## Міністерство освіти і науки України

## Запорізький національний технічний університет

кафедра програмних засобів

# звіт

з лабораторної роботи № 3

з дисципліни «Якість програмного забезпечення» на тему:

# «ОБЧИСЛЕННЯ МЕТРИК РОЗМІРУ ТА СКЛАДНОСТІ ПРОГРАМ»

Виконав:

студент групи КНТ-115 Е. В. Гунько

Прийняли:

Н.О. Миронова

2018

## Мета роботи:

Вивчити основні метрики коду програмних систем та реалізувати їх із застосуванням мови С++ та Qt.

## Завдання до роботи:

Реализувати на мові C++ з використанням крос-платформного інструментарію Qt програму розрахунку метрик:

* кількість рядків коду;
* кількість пустих рядків;
* кількість рядків, що містять коментарі;
* кількість рядків, що містять вихідний код і коментарі;
* відсоток коментарів;
* словник операторів (кількість унікальних операторів програми, також символи-роздільники, імена процедур і знаки операцій);
* словник операндів (кількість унікальних операндів програми);
* загальна кількість операторів в програмі;
* загальна кількість операндів в програмі;
* довжина програми;
* обсяг програми;
* цикломатична складність;
* максимальний рівень вкладеності і довжина програми.

## Хід роботи:

***Код класу mainwindow.cpp:***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) : QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_SLOC\_clicked()

{

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

int n = str.count("\n");

ui->label->setText(QString::number(n+1));

}

void MainWindow::on\_EmptyLines\_clicked()

{

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

int n = str.count("\n\n");

ui->label\_2->setText(QString::number(n));

}

void MainWindow::on\_Comments\_clicked()

{

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

int n = str.count("\n//");

ui->label\_3->setText(QString::number(n));

int k = str.count("\n");

ui->label\_5->setText(QString::number(((double) n)/(k+1)\*100)+"%");

}

void MainWindow::on\_OnlyCode\_clicked()

{

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

int n = str.count("\n");

int k = str.count("\n//");

ui->label\_4->setText(QString::number(n-k-str.count("\n\n")));

}

void MainWindow::on\_VocOperandn2\_clicked()

{

ui->label\_6->setText(QString::number(GetVocOperands()));

}

int MainWindow::GetVocOperands()

{

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

QRegExp rx("([\\w\\']+)[\\s,.;]");

QStringList list;

int pos = 0;

while ((pos = rx.indexIn(str, pos)) != -1) {

list << rx.cap(1);

pos += rx.matchedLength();

}

list.removeDuplicates ();

QString types = "int void string double float iterator List const array for while do goto";

QStringList ltypes;

ltypes = types.split(" ");

QStringList::Iterator itt = ltypes.begin();

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

{

const QString& item = list[i];

itt = ltypes.begin();

while (itt != ltypes.end()){

if(item==\*itt)

list.removeAt(i);

++itt;}

if ((item.toFloat()!=0)) {list.removeAt(i);}

}

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

{

const QString& item = list[i];

if (item=="0") {list.removeAt(i);}

}

return list.count();

}

void MainWindow::on\_VocOperatn1\_clicked()

{

ui->label\_7->setText(QString::number(GetVocOperations()));

}

int MainWindow::GetVocOperationsTotal()

{

return GetR() + GetVocOperations();

}

int MainWindow::GetR()

{

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

QStringList list = str.split(" ");

QString types = "+ - = += ++ -- \* << >> < > != == || && &";

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

QStringList ltypes;

QStringList ress;

list.removeDuplicates ();

ltypes = types.split(" ");

QStringList::Iterator itt = ltypes.begin();

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

{

const QString& item = list[i];

itt = ltypes.begin();

while (itt != ltypes.end()){

if(item==\*itt){

// list.removeAt(i);

ress << item;

}

++itt;

}

}

int r=0;

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

{

const QString& item = lst[i];

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

}

return r;

}

int MainWindow::GetVocOperations()

{

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

QStringList list = str.split(" ");

QString types = "+ - = += ++ -- \* << >> < > != == || && &";

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

QStringList ltypes;

QStringList ress;

list.removeDuplicates ();

ltypes = types.split(" ");

QStringList::Iterator itt = ltypes.begin();

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

{

const QString& item = list[i];

itt = ltypes.begin();

while (itt != ltypes.end()){

if(item==\*itt){

// list.removeAt(i);

ress << item;

}

++itt;

}

}

return ress.count();

}

void MainWindow::on\_AllOperatN1\_clicked()

{

ui->label\_8->setText(QString::number(GetAllOperations() + GetR()));

}

int MainWindow::GetAllOperations()

{

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

QStringList list = str.split(" ");

QString types = "+ - = += ++ -- \* << >> < > != == || && &";

QStringList ltypes;

QStringList ress;

ltypes = types.split(" ");

QStringList::Iterator itt = ltypes.begin();

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

{

const QString& item = list[i];

itt = ltypes.begin();

while (itt != ltypes.end()){

if(item==\*itt){

ress << item;

}

++itt;

}

}

return ress.count();

}

void MainWindow::on\_AllOperandN2\_clicked()

{

ui->label\_9->setText(QString::number(GetAllOperands()));

}

int MainWindow::GetAllOperands()

{

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

QRegExp rx("([\\w\\']+)[\\s,.;]");

QStringList list;

int pos = 0;

while ((pos = rx.indexIn(str, pos)) != -1) {

list << rx.cap(1);

pos += rx.matchedLength();

}

QString types = "int void string double float iterator List const array for while do goto";

QStringList ltypes;

ltypes = types.split(" ");

QStringList::Iterator itt = ltypes.begin();

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

{

const QString& item = list[i];

itt = ltypes.begin();

while (itt != ltypes.end()){

if(item==\*itt)

list.removeAt(i);

++itt;}

if ((item.toFloat()!=0)) {list.removeAt(i);}

}

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

{

const QString& item = list[i];

if (item=="0") {list.removeAt(i);}

}

return list.count();

}

void MainWindow::on\_Cyclomatic\_clicked()

{

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

QStringList list = str.split(QRegExp("(for|while)"), QString::SkipEmptyParts);

int s=0;

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

{

const QString& item = list[i];

if ((item.count("{")>=item.count("}"))&&(item.count("}")!=0)) {s=s+1; /\*ui->plainTextEdit->insertPlainText(\*it+"+++");\*/}

}

ui->label\_10->setText(QString::number(s+str.count("if(")));

}

void MainWindow::on\_Voc\_clicked()

{

ui->label\_11->setText(QString::number(GetVocOperands() + GetVocOperations() + GetR()));

}

void MainWindow::on\_N\_clicked()

{

ui->label\_12->setText(QString::number(GetLength()));

}

int MainWindow::GetLength()

{

return GetAllOperands() + GetAllOperations() + GetR();

}

void MainWindow::on\_V\_clicked()

{

double q = GetLength()\*qLn((GetVocOperands() + GetVocOperations() + GetR()))/qLn(2);

ui->label\_13->setText(QString::number(q));

}

void MainWindow::on\_CI\_clicked()

{

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

int n = str.count("if(");

ui->label\_14->setText(QString::number(n));

}

## Результати програми:

Результат роботи розробленої програми наведено на рисунку 1.

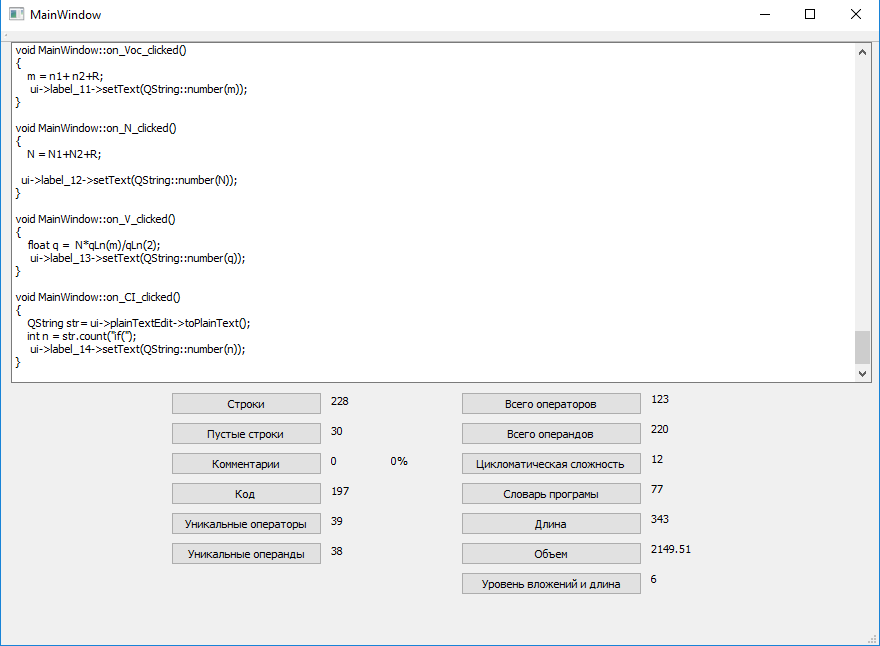


Рисунок 1 – Результат роботи розробленої програми

## Висновки:

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