****

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 6**

з дисципліни “ Основи програмування ”

тема: “Віконні програми”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Виконав  студент I курсу  групи КП-61  Cвинарчук Максим Владиславович  (*прізвище, ім’я, по батькові*)  варіант № 23 | |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2017 р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) | |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  |   Київ 2017 | | | Сумарний бал:   |  | | --- | |  | |

**Мета роботи**

**Мета**

Навчитися створювати віконні програми в ОС Linux за допомогою фреймворку Qt та редактора Qt Creator.

**Постановка завдання**

**Завдання №1. Типи даних**

Описати С++ клас Мова програмування. Розділити інтерфейс і реалізацію класy між файлами TypeName.h та typename.cpp (де замість typename - Мова програмування англійською мовою).

Всі поля даних класів мають бути приватними, для доступу до них використати методи геттери і сеттери. Створити декілька конструкторів класу.

**Завдання №2. Головне вікно**

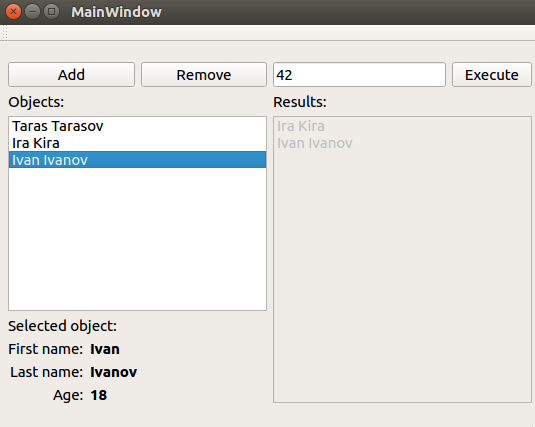
За допомогою стандартних колекцій мови С++ ініціалізувати список екземплярів класу *Мова програмування*. На головному вікні розмістити список елементів, що відображає створений список екземплярів користувачу (у рядках списку відображати не всю інформацію, а лише назви екземплярів) і форму (пари текстовий опис поля і текстове значення значення поля) у якій відображати значення обраного елемента списку.

Поруч зі списком розмістити на вікні кнопки Add, Remove та Execute, поле для вводу тексту та ще один список для результатів:

* Нажаття на Add викликає додаткове вікно для створення нового елемента.
* Нажаття на Remove видаляє обраний елемент зі списку (тому дана кнопка має бути активною тільки коли якийсь елемент списку обраний).
* Нажання та Compute отримує із поля для вводу значення X та виконує дію: *Знайти всі мови програмування створені до року Х*.

Результатуючі елементи занести у список результатів.

Приблизний вид головного вікна:



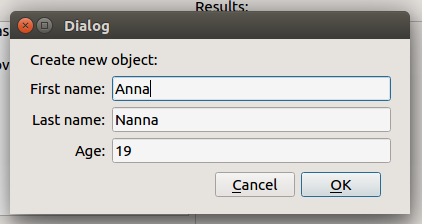
**Завдання №3. Діалог створення нових елементів**

Реалізувати діалог для створення нових екземплярів класу *Мова програмування*, яке викликається за допомогою кнопки Add із головного вікна.

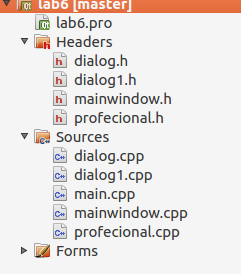
Діалог має містити форму (пари текстовий опис поля і поле для вводу значення) для вводу значень полів нового екземпляра та дві кнопки OK та Cancel:

* Cancel відміняє створення нового елемента та закриває діалог.
* OK створює новий екземпляр класу із введеними користувачем значеннями, закриває діалог і додає новий елемент у список головного вікна.

Приблизний вид діалогу створення нового елемента:



**Структура програми**

****

**Тексти коду програм**

|  |
| --- |
| main.cpp |
| **#include** "mainwindow.h"  **#include** <QApplication>  **int** **main**(**int** argc, **char** **\***argv[])  {  QApplication a(argc, argv);  MainWindow w;  w.show();    **return** a.exec();  } |

|  |
| --- |
| proglang.h |
| **#ifndef PROGLANG\_H**  **#define PROGLANG\_H**    **#include** <iostream>  **#include** <QString>  **#include** <QMetaType>    class Info {  QString \_majorImplementation;  **float** \_satisfactionIndex;  public:  Info();  **~**Info();  **void** **setMajorImpl**(QString majorImplementation);  **void** **setSatisfactionIndex**(**float** satisfactionIndex);  QString **getMajorImpl**();  **float** **getSatisfactionIndex**();  };    class ProgLang  {  QString \_fullname;  **int** \_dateOfOccurrence;  **float** \_marketShare;  Info \_data;    public:  ProgLang();  ProgLang(QString fullname, **int** dateOfOccurrence, **float** marketShare, QString majorImplementation, **float** satisfactionIndex);  **~**ProgLang();  **void** **setFullname**(QString fullname);  **void** **setDateOfOccurrence**(**int** dateOfOccurrence);  **void** **setMarketShare**(**float** marketShare);  QString **getFullname**();  **int** **getDateOfOccurrence**();  **float** **getMarketShare**();  Info **getData**();  };    Q\_DECLARE\_METATYPE(ProgLang)    **#endif** *// PROGLANG\_H* |
| proglang.cpp |
| **#include** "proglang.h"  **#include** <iostream>  Info**::**Info()  {  \_majorImplementation **=** "";  \_satisfactionIndex **=** 0;  }    Info**::~**Info() {}    **void** Info**::**setMajorImpl(QString majorImplementation)  {  \_majorImplementation **=** majorImplementation;  }    **void** Info**::**setSatisfactionIndex(**float** satisfactionIndex)  {  \_satisfactionIndex **=** satisfactionIndex;  }    QString Info**::**getMajorImpl()  {  **return** \_majorImplementation;  }    **float** Info**::**getSatisfactionIndex()  {  **return** \_satisfactionIndex;  }    ProgLang**::**ProgLang()  {  \_fullname **=** "";  \_dateOfOccurrence **=** 0;  \_marketShare **=** 0;  }    ProgLang**::**ProgLang(QString fullname, **int** dateOfOccurrence, **float** marketShare,  QString majorImplementation, **float** satisfactionIndex)  {  \_fullname **=** fullname;  \_dateOfOccurrence **=** dateOfOccurrence;  \_marketShare **=** marketShare;  \_data.setMajorImpl(majorImplementation);  \_data.setSatisfactionIndex(satisfactionIndex);  }    ProgLang**::~**ProgLang() {}    **void** ProgLang**::**setFullname(QString fullname)  {  \_fullname **=** fullname;  }    **void** ProgLang**::**setDateOfOccurrence(**int** dateOfOccurrence)  {  \_dateOfOccurrence **=** dateOfOccurrence;  }    **void** ProgLang**::**setMarketShare(**float** marketShare)  {  \_marketShare **=** marketShare;  }    QString ProgLang**::**getFullname()  {  **return** \_fullname;  }    **int** ProgLang**::**getDateOfOccurrence()  {  **return** \_dateOfOccurrence;  }    **float** ProgLang**::**getMarketShare()  {  **return** \_marketShare;  }    Info ProgLang**::**getData() {  **return** \_data;  } |
| mainwindow.h |
| **#ifndef MAINWINDOW\_H**  **#define MAINWINDOW\_H**    **#include** <QMainWindow>  **#include** <proglang.h>  **#include** <vector>  **#include** <dialog.h>  namespace Ui {  class MainWindow;  }    class MainWindow : public QMainWindow  {  Q\_OBJECT    public:  explicit MainWindow(QWidget **\***parent **=** 0);  **~**MainWindow();  **void** **addProgLang**(ProgLang **\*** self);    private slots:  **void** closeEvent(QCloseEvent **\*** event);    **void** **on\_listWidget\_itemSelectionChanged**();    **void** **on\_pushButton\_3\_clicked**();    **void** **on\_pushButton\_4\_clicked**();    **void** **on\_pushButton\_2\_clicked**();    private:  Ui**::**MainWindow **\***ui;    std**::**vector**<**ProgLang **\*>** languages;  };    **#endif** *// MAINWINDOW\_H* |
| mainwindow.cpp |
| **#include** "mainwindow.h"  **#include** <QDebug>  **#include** <QCloseEvent>  **#include** <QApplication>  **#include** <QListWidget>  **#include** <dialog.h>  **#include** "ui\_mainwindow.h"    MainWindow**::**MainWindow(QWidget **\***parent) **:**  QMainWindow(parent),  ui(**new** Ui**::**MainWindow)  {  ui**->**setupUi(**this**);  languages.push\_back(**new** ProgLang("C", 1972, 1.3, "gcc", 0.33));  languages.push\_back(**new** ProgLang("C++", 1983, 6.4, "g++", 0.61));  languages.push\_back(**new** ProgLang("C#", 2001, 15.4, ".NET\_Framework", 0.80));  languages.push\_back(**new** ProgLang("Java", 1995, 26.3, "JVM", 0.78));  languages.push\_back(**new** ProgLang("JavaScript", 1995, 13.8, "SpiderMonkey", 0.71));  ui**->**listWidget**->**setIconSize(QSize(32, 32));  foreach (ProgLang **\*** cur, languages) {  QVariant qVariant;  qVariant.setValue(**\***cur);  QListWidgetItem **\***qLanguagesListItem **=** **new** QListWidgetItem();  qLanguagesListItem**->**setText(cur**->**getFullname());  qLanguagesListItem**->**setIcon(QIcon("/home/valzavator/progbase2/labs/lab6/images/" **+** cur**->**getFullname() **+** ".jpeg"));  qLanguagesListItem**->**setData(Qt**::**UserRole, qVariant);  ui**->**listWidget**->**addItem(qLanguagesListItem);  }    }    MainWindow**::~**MainWindow()  {  **delete** ui;  }    **void** MainWindow**::**closeEvent(QCloseEvent **\***event)  {  *// use this to cleanup allocated eventmemory*    languages.erase(languages.begin(), languages.end());  qDebug() **<<** "Window closed!";  event**->**accept();  }    **void** MainWindow**::**on\_listWidget\_itemSelectionChanged()  {  qDebug() **<<** "changed!";  **auto** items **=** ui**->**listWidget**->**selectedItems();  **auto** hasSelected **=** items.size() **>** 0;  **if** (hasSelected) {  ui**->**pushButton\_3**->**setEnabled(true);  ProgLang lang **=** items[0]**->**data(Qt**::**UserRole).value**<**ProgLang**>**();  ui**->**label\_2**->**setText(lang.getFullname());  ui**->**label\_3**->**setText(QString**::**number(lang.getDateOfOccurrence()));  ui**->**label\_4**->**setText(QString**::**number(lang.getMarketShare(), 'f', 2));  ui**->**label\_5**->**setText(lang.getData().getMajorImpl());  ui**->**label\_6**->**setText(QString**::**number(lang.getData().getSatisfactionIndex(), 'f', 2));  } **else** {  ui**->**pushButton\_3**->**setEnabled(false);  ui**->**label\_2**->**setText("");  ui**->**label\_3**->**setText("");;  ui**->**label\_4**->**setText("");;  ui**->**label\_5**->**setText("");;  ui**->**label\_6**->**setText("");;  }  }    **void** MainWindow**::**on\_pushButton\_3\_clicked()  {  QList**<**QListWidgetItem**\*>** items **=** ui**->**listWidget**->**selectedItems();    foreach(QListWidgetItem **\*** item, items)  {  **int** toDelete **=** ui**->**listWidget**->**row(item);  **delete** ui**->**listWidget**->**takeItem(ui**->**listWidget**->**row(item));  languages.erase(languages.begin() **+** toDelete);  }    }    **void** MainWindow**::**on\_pushButton\_4\_clicked()  {  ui**->**listWidget\_2**->**clear();  **int** year **=** ui**->**spinBox**->**value();  **int** count **=** languages.size();  **for** (**int** i **=** 0; i **<** count; i**++**) {  **if** (languages[i]**->**getDateOfOccurrence() **<** year) {  QVariant qVariant;  qVariant.setValue(**\***languages[i]);  QListWidgetItem **\***qLanguagesListItem **=** **new** QListWidgetItem();  qLanguagesListItem**->**setText(languages[i]**->**getFullname());  ui**->**listWidget\_2**->**setIconSize(QSize(32, 32));  qLanguagesListItem**->**setText(languages[i]**->**getFullname());  qLanguagesListItem**->**setIcon(QIcon("/home/valzavator/progbase2/labs/lab6/images/" **+** languages[i]**->**getFullname() **+** ".jpeg"));  qLanguagesListItem**->**setData(Qt**::**UserRole, qVariant);  ui**->**listWidget\_2**->**addItem(qLanguagesListItem);  }  }  }    **void** MainWindow**::**on\_pushButton\_2\_clicked()  {  Dialog **\*** dialog **=** **new** Dialog();  dialog**->**setWindowTitle("Add new language");  **if** (dialog**->**exec()) {  qDebug() **<<** "Accepted!";  ProgLang **\*** newLang **=** dialog**->**getData();  languages.push\_back(newLang);  QVariant qVariant;  qVariant.setValue(**\***newLang);  QListWidgetItem **\***qLanguagesListItem **=** **new** QListWidgetItem();  ui**->**listWidget**->**setIconSize(QSize(32, 32));  qLanguagesListItem**->**setText(newLang**->**getFullname());  qLanguagesListItem**->**setIcon(QIcon("/home/valzavator/progbase2/labs/lab6/images/" **+** newLang**->**getFullname() **+** ".jpeg"));  qLanguagesListItem**->**setData(Qt**::**UserRole, qVariant);  ui**->**listWidget**->**addItem(qLanguagesListItem);  } **else** {  qDebug() **<<** "Rejected!";  }  } |
| dialog.h |
| **#ifndef DIALOG\_H**  **#define DIALOG\_H**    **#include** <QDialog>  **#include** <proglang.h>    namespace Ui {  class Dialog;  }    class Dialog : public QDialog  {  Q\_OBJECT    public:  explicit Dialog(QWidget **\***parent **=** 0);  Dialog(ProgLang **\*** data);  **~**Dialog();  ProgLang **\*** **getData**();    private slots:  **void** closeEvent(QCloseEvent **\*** event);    **void** **on\_buttonBox\_accepted**();    signals:  *// void sendData(QString fullname, int year, float marketShare, QString majorImpl,*  *// float satisfactionIndex );*  **void** sendData(ProgLang **\*** newLang);    private:  Ui**::**Dialog **\***ui;    ProgLang **\*** \_data;  };    **#endif** *// DIALOG\_H* |
| dialog.cpp |
| **#include** "dialog.h"  **#include** "ui\_dialog.h"  **#include** <QCloseEvent>  **#include** <QDebug>  **#include** <proglang.h>  **#include** <mainwindow.h>    Dialog**::**Dialog(QWidget **\***parent) **:**  QDialog(parent),  ui(**new** Ui**::**Dialog)  {  ui**->**setupUi(**this**);    }    Dialog**::~**Dialog()  {  **delete** ui;  }    **void** Dialog**::**closeEvent(QCloseEvent **\***event)  {  *// use this to cleanup allocated eventmemory*      qDebug() **<<** "Dialog closed!";  event**->**accept();  }    **void** Dialog**::**on\_buttonBox\_accepted()  {  QString fullname **=** ui**->**fullnameLineEdit**->**text();  **int** year **=** ui**->**dateOfOccurrenceSpinBox**->**value();  **float** marketShare **=** ui**->**marketShareDoubleSpinBox**->**value();  QString majorImpl **=** ui**->**majorImplementationLineEdit**->**text();  **float** satisfactionIndex **=** ui**->**satisfactionIndexDoubleSpinBox**->**value();  ProgLang **\*** newLang **=** **new** ProgLang(fullname, year, marketShare, majorImpl, satisfactionIndex);  \_data **=** newLang;  }    Dialog**::**Dialog(ProgLang **\*** data) {  data **=** data;  }    ProgLang **\*** Dialog**::**getData() {  **return** \_data;  } |

**Приклад результатів**

**Завдання 1**

Клас*«*Мова програмування» містить 4 приватні поля:

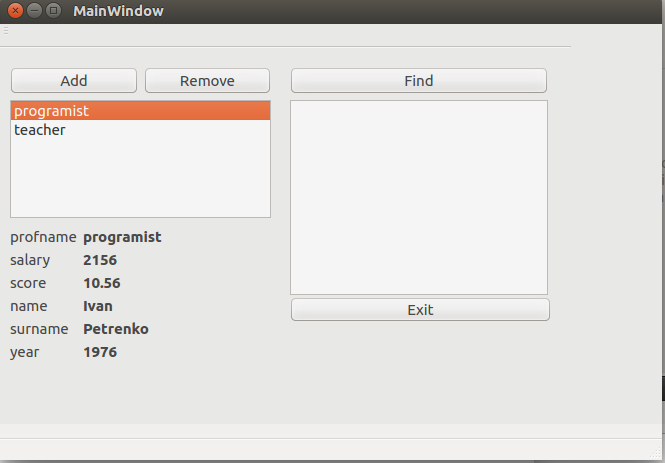
1. QString \_fullname – назва.
2. int \_dateOfOccurrence – дата створення.
3. float \_marketShare – доля ринку.
4. Info \_data – клас «Info», який у свою чергу містить два поля(Головна реалізація та індекс задоволеності).

**Завдання 2**

На головному вікні розміщений список елементів, що відображає створений список екземплярів користувача(у рядках списку відображена назва мови програмування та картинка) і форма (пари текстовий опис поля і текстове значення значення поля) у якій відображаються значення обраного елемента списку.

Поруч зі списком розміщені на вікні кнопки Add, Remove та Execute, поле для вводу тексту та ще один список для результатів:

* Нажаття на Add викликає додаткове вікно для створення нового елемента.
* Нажаття на Remove видаляє обраний елемент зі списку (тому дана кнопка має бути активною тільки коли якийсь елемент списку обраний).
* Нажання та Compute отримує із поля для вводу значення X та виконує дію: *Знайти всі мови програмування створені до року Х*.



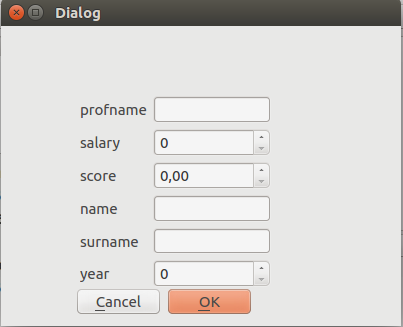
**Завдання 3**

Діалог викликається за допомогою кнопки Add із головного вікна.

Діалог містити форму (пари текстовий опис поля і поле для вводу значення) для вводу значень полів нового екземпляра та дві кнопки OK та Cancel:

* Cancel відміняє створення нового елемента та закриває діалог.
* OK створює новий екземпляр класу із введеними користувачем значеннями, закриває діалог і додає новий елемент у список головного вікна.

Вид діалогу створення нового елемента:

****

**Висновок**

Виконавши дану лабораторну роботу були отримані навички роботи з класами в С++.

Я навчився створювати віконні програми в ОС Linux за допомогою фреймворку Qt та редактора Qt Creator.

Були отримані навички роботи з QDialog, QListWidget, Label, Button, та іншими елементами віконних програм, їх програмування та взаємодія між собою.