# 3 Создание программной системы c Qt интерфейсом на С++

***Задание:***

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

Сведения о программных продуктах представлены названием каждого из них, датой выпуска, компанией-разработчиком, требуемым объемом оперативной памяти. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Определить, какие из программных продуктов требуют памяти не более указанного объема.

2. Определить, какие программные продукты созданы данным разработчиком за указанный период.

3. Показать самые требовательные к памяти программы каждого разработчика.

4. Построить график изменения количества выпущенных программ по годам.

3.1 Объектная декомпозиция приложения

При проектировании программного продукта были выделены следующие объекты предметной области:

- окно;

- таблица;

- кнопка;

- холст;

- продукт.

Диаграмма объектов представлена на рисунке 1.



Рисунок 1 – окно программы; Окно 2 – добавление продукта; Окно 3 – удаление продукта; Окно 4 – показать продукты; Окно 5 – график выпущенных программ по годам.

3.2 Разработка форм интерфейса

Разрабатываемые формы интерфейса должны обеспечивать пользователю возможность выполнения заданных функций.

Форма 1 предназначена для запуска программы, соответственно она должна включать кнопки, обеспечивающие добавление, удаление и показ данных, а также выхода из всей программы (рисунок 2).

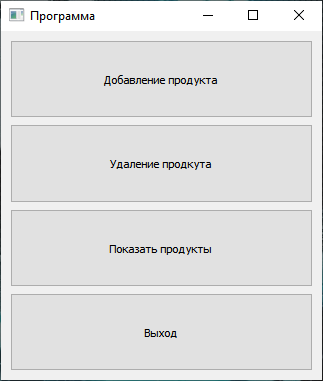


Рисунок 2 – Внешний вид формы «Программа»

Форма 2 предназначена для добавления данных, соответственно она должна включать поля для добавления данных и кнопку, обеспечивающую их ввод в программу (рисунок 3).

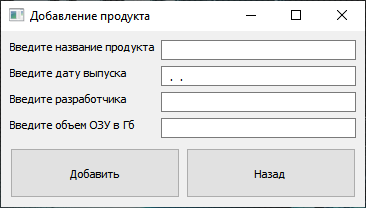


Рисунок 3 – Внешний вид формы «Добавление данных»

Форма 3 предназначена показа данных и выполнения ряда команд, соответственно она должна включать поля для добавления данных и кнопки для их ввода.

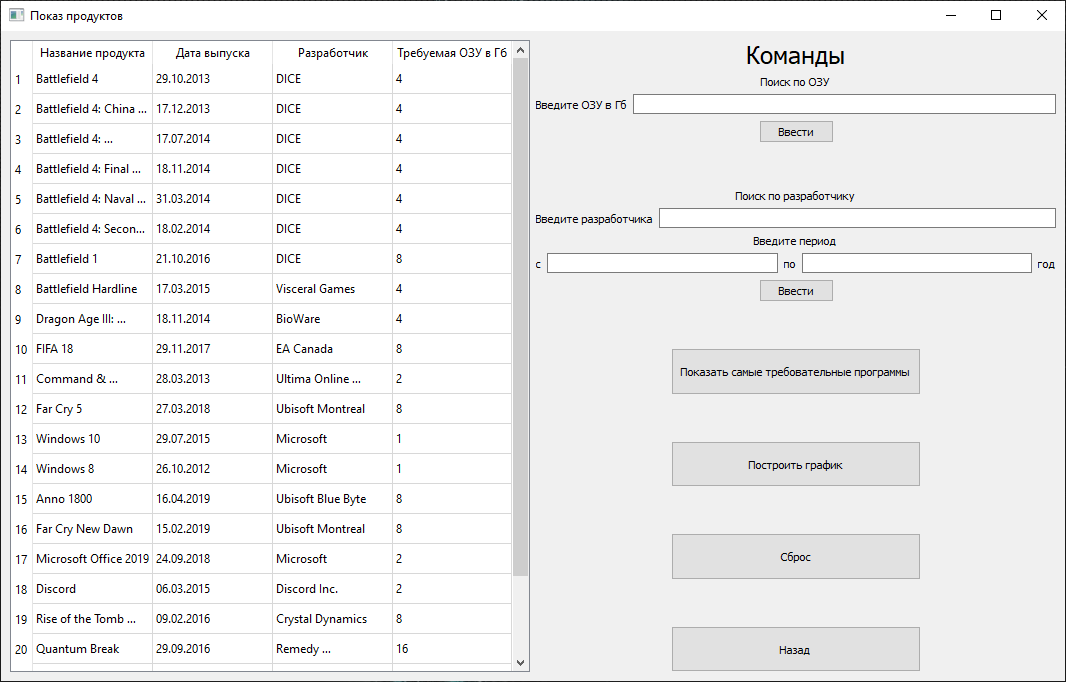


Рисунок 4 – Внешний вид формы «Показать продукты»

Остальные формы были спроектированы аналогично.

3.3 Разработка диаграммы состояний интерфейса

Диаграмма состояний интерфейса показывает возможные варианты переключения форм интерфейса (рисунок 5).



Рисунок 5 – Диаграмма состояний интерфейса

Где: С1 – запуск программы; С2 – выход из программы с помощью стандартной кнопки закрытия окна; C3 – нажатие кнопки “Добавление продукта”; C4 – нажатие кнопки “Удаление продукта”; C5 – нажатие кнопки “Показать продукты”; С6 – ввод данных и нажатие кнопки “Добавить”; С7 – ввод данных и нажатие кнопки “Удалить”; С8, C9, C10, C11 и C12 – соответственно команда поиска по ОЗУ, команда поиска по разработчику за промежуток, команда нахождение программ требовательных разработчиков, команда построения графика изменения количества выпущенных программ по годам и команда сброса данных в таблице; C13, C14, 15 – нажатие кнопки “Назад”.

Аналогично разрабатываются диаграммы состояний интерфейса для отдельных форм.

3.4 Разработка диаграммы классов интерфейсной и предметной областей приложения

Диаграмма классов интерфейсной и предметной областей показывает связи между классами (рисунок 6-13).



Рисунок 6 «Разработанный класс product»



Рисунок 7 «Диаграмма классов приложения»

3.5 Разработка диаграммы последовательности действий выполнения операции

Диаграмма последовательностей действий позволяет уточнить порядок выполнения фрагментов операции различными объектами (рисунок 8).

Рисунок 8 – Диаграмма последовательности действий выполнения операции добавления элементов

3.6 Разработка кода приложения

Уточнение поведения объектов во время выполнения программы позволило разработать код программы.

Ниже в качестве примера приведен код заголовочного файла chartwindow.h.

#ifndef CHARTWINDOW\_H

#define CHARTWINDOW\_H

#include <QMainWindow>

#include <QtCharts/QtCharts>

#include <product.h>

QT\_CHARTS\_USE\_NAMESPACE

using namespace QtCharts;

namespace Ui {

class ChartWindow;

}

class ChartWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit ChartWindow(QWidget \*parent = nullptr);

~*ChartWindow*();

private:

Ui::ChartWindow \*ui;

};

#endif // CHARTWINDOW\_H

Ниже приведен также код исполняемого файла chartwindow.cpp.

*#include "chartwindow.h"*

*#include "ui\_chartwindow.h"*

*ChartWindow::ChartWindow(QWidget \*parent) :*

*QMainWindow(parent),*

*ui(new Ui::ChartWindow)*

*{*

*ui->setupUi(this);*

*QChart \*chart = new QChart();*

*QSplineSeries \*series = new QSplineSeries();*

*product pr;*

*std::ifstream fis("file.dat", std::ios::binary);*

*QVector<product> vect;*

*while (fis.read(reinterpret\_cast<char\*>(&pr), sizeof(pr)))*

*{*

*vect.push\_back(pr);*

*}*

*fis.close();*

*int min,max;*

*min = max = atoi(vect[0].Get\_Release()+6);*

*for (int i=0; i < vect.size(); i++)*

*{*

*if (atoi(vect[i].Get\_Release()+6) > max)*

*max = atoi(vect[i].Get\_Release()+6);*

*if (atoi(vect[i].Get\_Release()+6) < min)*

Продолжение кода исполняемого файла chartwindow.cpp.

*min = atoi(vect[i].Get\_Release()+6);*

*}*

*int k=0;*

*for (int i = min; i <= max; i++)*

*{*

*int count=0;*

*for (int j=0; j < vect.size(); j++)*

*if (atoi(vect[j].Get\_Release()+6) == i)*

*count++;*

*series->append(i, count);*

*k++;*

*}*

*chart->addSeries(series);*

*chart->setTitle("График изменения количества выпущенных программ по годам");*

*QDateTimeAxis \*axisX = new QDateTimeAxis;*

*axisX->setTickCount(k);*

*axisX->setFormat("yyyy");*

*QDateTime minDate;*

*minDate.setDate(QDate(min, 1 , 1));*

*minDate.setTime(QTime(0, 00));*

*QDateTime maxDate;*

*maxDate.setDate(QDate(max, 12 , 31));*

*maxDate.setTime(QTime(23, 59));*

*axisX->setRange(minDate, maxDate);*

*axisX->setTitleText("Год");*

*chart->addAxis(axisX, Qt::AlignBottom);*

*QValueAxis \*axisY = new QValueAxis;*

*axisY->setLabelFormat("%i");*

*axisY->setTitleText("Количество выпущенных программ");*

*chart->addAxis(axisY, Qt::AlignLeft);*

*series->attachAxis(axisY);*

*QChartView \*chartView = new QChartView(chart);*

*chartView->chart()->legend()->hide();*

*chartView->setRenderHint(QPainter::Antialiasing);*

*setCentralWidget(chartView);*

*}*

*ChartWindow::~ChartWindow()*

*{*

*delete ui;*

*}*

Аналогично был разработан остальной код приложения.

3.7 Тестирование приложения

Разработанное приложение позволяет выполнять функции, предусмотренные заданием (рисунки 9-11) .

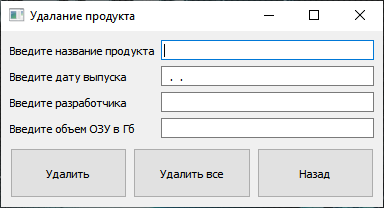


Рисунок 9 «Окно удаления элементов»

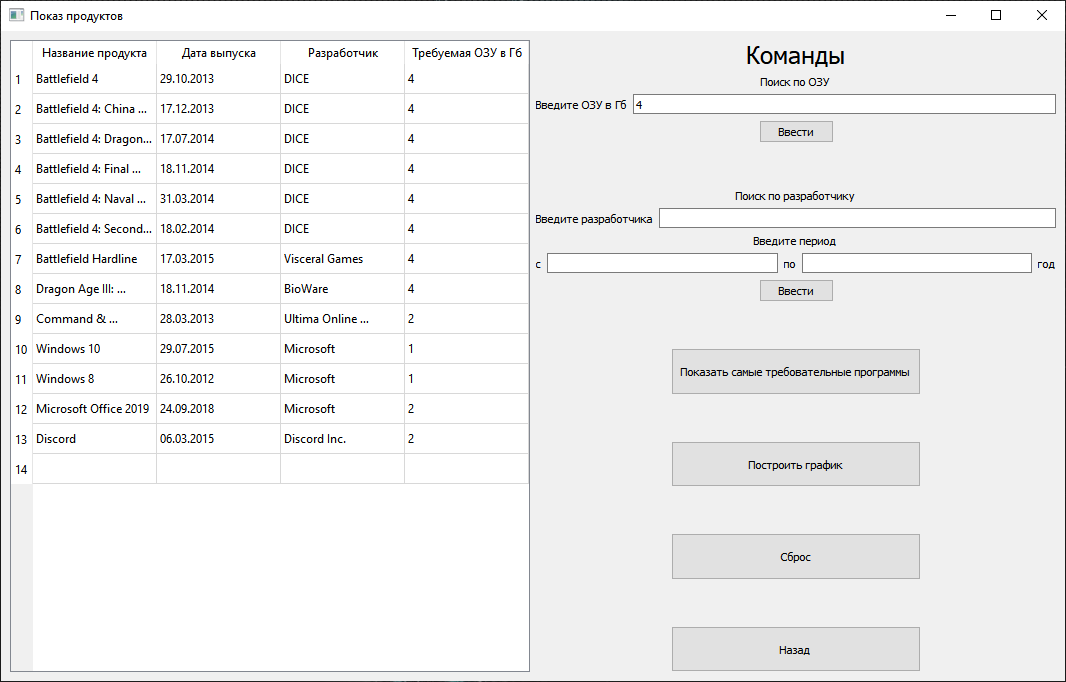


Рисунок 10 «Окно выполнения поиска продуктов по ОЗУ»

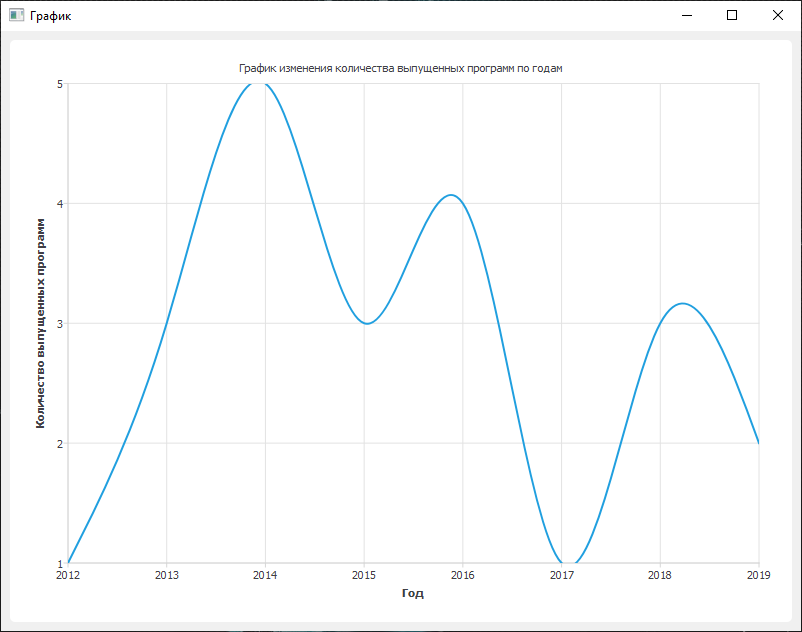


Рисунок 11 «Окно график изменения количества выпущенных программ по годам»

Вывод

При разработке приложения изучены средства разработки приложений с графическим интерфейсом на языке C++, используемые при объектном подходе, а также основные диаграммы, сопровождающие процесс разработки. При реализации проще оказалась разработка в Qt Creator, чем на Object Pascal, так как в нем давалось больше готовых инструментов для создания приложения.