|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

|  |  |
| --- | --- |
| Студент | Каташинский Кирилл Юрьевич |
|  | *Фамилия имя отчество* |

|  |  |
| --- | --- |
| Группа | ИУ6-24Б |

|  |  |
| --- | --- |
| Тип практики | Проектно-технологическая практика |

|  |  |
| --- | --- |
| Название  предприятия | НУК ИУ МГТУ им. Н.Э. Баумана |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | К.Ю. Каташинский |
|  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |
| Руководитель практики |  |  | А.М. Минитаева |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ6\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Пролетарский

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**ЗАДАНИЕ**

**на учебную практику**

по теме \_\_\_\_Проектирование и реализация программного обеспечения с использованием структурного и объектного подходов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы ИУ6-24Б

Каташинский Кирилл Юрьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тип практики Проектно-технологическая практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

***Техническое задание:***

**Задание 1. Создание программной системы на Object Pascal**

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.

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.

**Задание 2. Создание программной системы с элементарным интерфейсом консольного режима на С++**

Выполнить структурную декомпозицию, разработать структурную схему, содержащую не менее 3 подпрограмм, и алгоритмы этих подпрограмм. Реализовать на С++ в консольном режиме. Предусмотреть примитивный интерфейс типа меню, позволяющий выбирать нужную подпрограмму.

Разработать программу, которая выполняет простейшее исследование функций одной переменной. Реализовать следующие операции: ввод функции, преобразование ее во внутреннее представление (дерево), определение экстремумов функции на заданном отрезке, а также вывод результатов на экран.

**Задание 3. Создание программной системы c Qt интерфейсом на С++**

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.

***Оформление отчета по практике:***

Расчетно-пояснительная записка на 25-35 листах формата А4 должна включать титульный лист, задание (печатать с двух сторон), оглавление, введение, три главы и заключение. Отдельная глава по каждому заданию должна содержать анализ задания, требуемые чертежи, текст программы, результаты тестирования и выводы.

Дата выдачи задания « 07 » февраля 2020 г.

|  |  |  |  |
| --- | --- | --- | --- |
| **Руководитель практики** |  |  |  |
|  |  | (Подпись, дата) | (И.О. Фамилия) |
| **Студент** |  |  |  |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Оглавление**

[Введение 5](#_Toc38710045)

[1 Создание программной системы на Object Pascal 6](#_Toc38710046)

[1.1 Объектная декомпозиция приложения 6](#_Toc38710047)

[1.2 Разработка форм интерфейса 7](#_Toc38710048)

[1.3 Разработка диаграммы состояний интерфейса 9](#_Toc38710049)

[1.4 Разработка диаграммы классов интерфейсной и предметной областей приложения 10](#_Toc38710051)

[1.5 Разработка диаграммы последовательности действий выполнения операции 13](#_Toc38710053)

[1.6 Разработка кода приложения 13](#_Toc38710055)

[1.7 Тестирование приложения 16](#_Toc38710060)

[Вывод 17](#_Toc38710062)

[2 Создание программной системы с элементарным интерфейсом консольного режима на С++ 18](#_Toc38710064)

[2.1 Структурная декомпозиция программы 18](#_Toc38710066)

[2.2 Разработка схем алгоритмов программы 18](#_Toc38710070)

[2.3 Разработка кода программы 22](#_Toc38710071)

[2.4 Тестирование программы 28](#_Toc38710075)

[Вывод 30](#_Toc38710077)

[3 Создание программной системы c Qt интерфейсом на С++ 31](#_Toc38710079)

[3.1 Объектная декомпозиция приложения 31](#_Toc38710080)

[3.2 Разработка форм интерфейса 32](#_Toc38710081)

[3.3 Разработка диаграммы состояний интерфейса 34](#_Toc38710082)

[3.4 Разработка диаграммы классов интерфейсной и предметной областей приложения 35](#_Toc38710084)

[3.5 Разработка диаграммы последовательности действий выполнения операции 37](#_Toc38710086)

[3.6 Разработка кода приложения 37](#_Toc38710088)

[3.7 Тестирование приложения 39](#_Toc38710092)

[Вывод 41](#_Toc38710094)

[Заключение 42](#_Toc38710096)

[Список литературы 43](#_Toc38710097)

# Введение

***Целью*** учебнойпрактики является получение навыков создания небольших программных систем с оконными и консольными интерфейсами.

***Задачами*** практики являются:

− более глубокое изучение средств реализации проектов программ на одном из изучаемых универсальных языках программирования высокого уровня;

− овладение методикой и получение практических навыков проектирования небольших программных систем при структурном и объектном подходах;

− воспитание внимания, аккуратности, систематичности, а также формирование интереса к изучаемой профессиональной деятельности.

Выполнение практикума должно способствовать формированию и развитию следующих навыков и умений:

− выделение объектов предметной области, обобщение их в классы, определение связей между классами;

− проектирование эргономичного обеспечения информационных систем;

− разработка и отладка компонентов программных комплексов и систем с помощью современных автоматизированных средств проектирования;

− разработка проектной и эксплуатационной документации на программную и техническую продукцию;

− выполнение контроля разрабатываемых проектов и технической документации на соответствие стандартам и техническим требованиям;

− разработка интерфейсов «человек - ЭВМ».

# 1 Создание программной системы на Object Pascal

***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.

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.

1.1 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 – Диаграмма объектов приложения

Где: Окно 1 – главное меню; Окно 2 – добавление данных; Окно 3 – удаление данных; Окно 4 – показ данных; Окно 5 – поиск по ОЗУ; Окно 6 – поиск за промежуток; Окно 7 – график выпущенных программ по годам.

1.2 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).

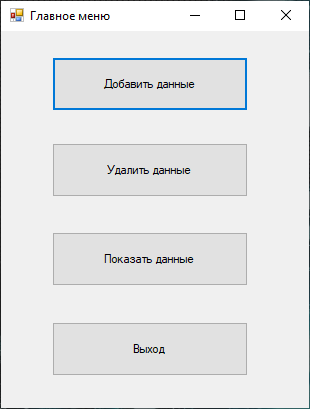


Рисунок 2 – Внешний вид формы «Главное меню»

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

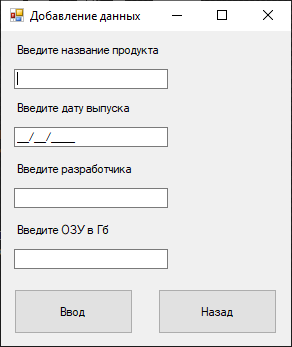


Рисунок 3 – Внешний вид формы «Добавление данных»

Форма 3 предназначена для поиска данных разработчика за указанный промежуток в годах, соответственно она должна включать поля для добавления данных и кнопку, обеспечивающую их ввод в программу (рисунок 4).

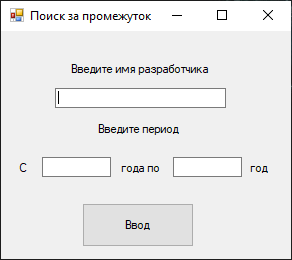


Рисунок 4 – Внешний вид формы «Поиск за промежуток»

Остальные формы были спроектированы аналогично.

1.3 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; С3 – нажатие кнопки «Добавить данные»; С4 – нажатие кнопки «Удалить данные»; С5 – нажатие кнопки «Показать данные»; С6 – нажатие кнопки «Назад»; С7 – нажатие кнопки «Поиск по ОЗУ»; С8 – нажатие кнопки «Поиск продуктов разработчика за промежуток»; С9 – нажатие кнопки «Показать самые требовательные программы»; С10 – нажатие кнопки «Построить график»; С11 – нажатие кнопки «Сброс»; С12 - выход из дополнительных окон с помощью стандартной кнопки закрытия окна.

Аналогично разрабатываются диаграммы состояний интерфейса для отдельных форм.

1.4 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 «Класс Form1 модуля unit1»



Рисунок 8 «Класс Form1 модуля unit2»



Рисунок 9 «Класс Form1 модуля unit3»



Рисунок 10 «Класс Form1 модуля unit4»



Рисунок 11 «Класс Form1 модуля unit5»



Рисунок 12 «Класс Form1 модуля unit6»



Рисунок 13 «Класс Form1 модуля unit7»

1.5 Разработка диаграммы последовательности действий выполнения операции

Диаграмма последовательностей действий позволяет уточнить порядок выполнения фрагментов операции различными объектами (рисунок 14).

Рисунок 14 – Диаграмма последовательности действий выполнения операции добавления данных

1.6 Разработка кода приложения

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

Ниже в качестве примера приведен код модуля 7.

*Unit Unit7;*

*interface*

*uses System, System.Drawing, System.Windows.Forms;*

*var*

*n:integer;*

*type*

*Form1 = class(Form)*

*procedure label1\_Click(sender: Object; e: EventArgs);*

*procedure label2\_Click(sender: Object; e: EventArgs);*

*{$region FormDesigner}*

*private*

*{$resource Unit8.Form1.resources}*

*label1: &Label;*

*label2: &Label;*

*pictureBox1: PictureBox;*

*{$include Unit8.Form1.inc}*

*{$endregion FormDesigner}*

*public*

*constructor(num:integer);*

*begin*

*InitializeComponent;*

*n:=num;*

*make\_function();*

*end;*

*procedure make\_function();*

*end;*

*function count(y:integer):integer;*

*implementation*

*uses mine;*

*var*

*graph:graphics;*

*bmp:bitmap;*

*myFont:font := new Font('Times New Roman', 11);*

*brush1:solidbrush:=new SolidBrush(color.black);*

*p:point;*

*pen1:pen;*

*procedure form1.make\_function();*

*var*

*i,min,max,move:integer;*

*begin*

*pen1:=new Pen(color.black);*

*p:=new Point(0,picturebox1.height-20);*

*bmp:=new Bitmap(picturebox1.Width,picturebox1.height);*

*graph := Graphics.fromimage(bmp);*

Продолжение кода модуля 7

*pictureBox1.Image := bmp;*

*i:=0;*

*pen1.DashStyle:=system.Drawing.drawing2d.DashStyle.dash;*

*while i<=20 do begin*

*graph.DrawString(inttostr(i),myfont,brush1,p.X,p.y-7);*

*i+=1;*

*p.Y-=24;*

*graph.DrawLine(pen1,20,p.y,picturebox1.width,p.y);*

*end;*

*pen1.DashStyle:=system.Drawing.drawing2d.DashStyle.solid;*

*graph.DrawLine(pen1,20,picturebox1.height-20,20,0);*

*graph.DrawLine(pen1,20,picturebox1.height-20,picturebox1.width,picturebox1.height-20);*

*min:=products[1].Get\_release\_date.year;*

*max:=products[1].Get\_release\_date.year;*

*for i:=1 to n do begin*

*if products[i].Get\_release\_date.year<min then*

*min:=products[i].Get\_release\_date.year;*

*if products[i].Get\_release\_date.year>max then*

*max:=products[i].Get\_release\_date.year;*

*end;*

*if max=min then begin*

*messageBox.Show('Одинаковые данные!','Ошибка',MessageBoxButtons.OK,MessageBoxIcon.Error);*

*exit;*

*end;*

*move:=round(picturebox1.width/(max-min))-15;*

*p:=new Point(45,picturebox1.height-15);*

*pen1.Color:=color.Blue;*

*for i:=min to max do begin*

*brush1.Color:=color.blue;*

*graph.DrawRectangle(pen1,p.x,p.Y-5-24\*count(i),30,24\*count(i));*

*graph.FillRectangle(brush1,p.x,p.Y-5-24\*count(i),30,24\*count(i));*

*brush1.Color:=color.Black;*

*graph.DrawString(inttostr(i),myfont,brush1,p);*

*p.x+=move;*

*end;*

*end;*

*function count(y:integer):integer;*

*var*

*i:integer;*

*begin*

*result:=0;*

*for i:=1 to n do*

*if products[i].Get\_release\_date.year=y then result+=1;*

*end;*

*end.*

Аналогично был разработан остальной код приложения.

1.7 Тестирование приложения

Разработанное приложение позволяет выполнять функции, предусмотренные заданием (рисунки 15-17) .

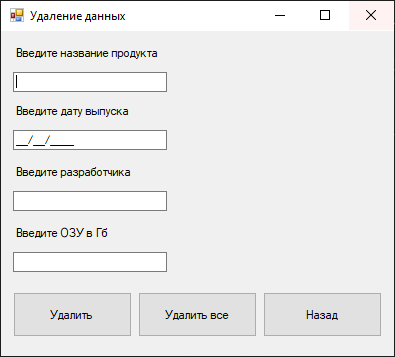


Рисунок 15 «Окно удаления данных»

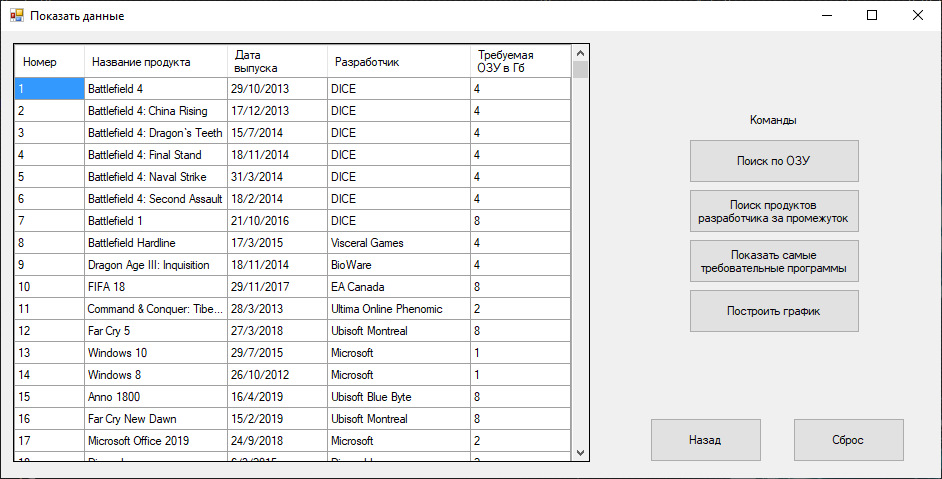


Рисунок 16 «Окно показать данные»

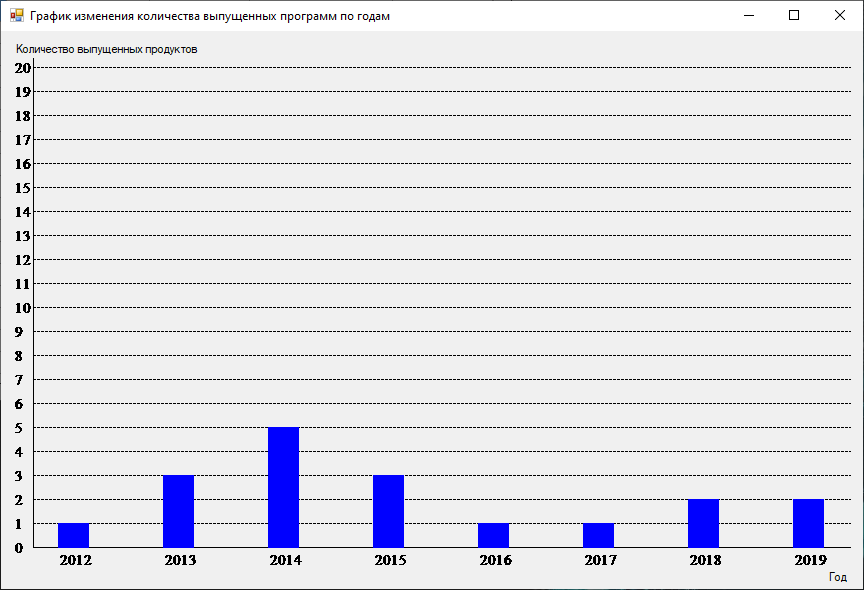


Рисунок 17 «Окно график изменения количества выпущенных программ по годам»

Вывод

При разработке приложения изучены средства разработки приложений с графическим интерфейсом на языке Object Pascal, используемые при объектном подходе, а также основные диаграммы, сопровождающие процесс разработки.

# 2 Создание программной системы с элементарным интерфейсом консольного режима на С++

***Задание:***

Выполнить структурную декомпозицию, разработать структурную схему, содержащую не менее 3 подпрограмм, и алгоритмы этих подпрограмм. Реализовать на С++ в консольном режиме. Предусмотреть примитивный интерфейс типа меню, позволяющий выбирать нужную подпрограмму.

Разработать программу, которая выполняет простейшее исследование функций одной переменной. Реализовать следующие операции: ввод функции, преобразование ее во внутреннее представление (дерево), определение экстремумов функции на заданном отрезке, а также вывод результатов на экран.

2.1 Структурная декомпозиция программы

При создании программы процесс разработки разделился на следующие части:

определение наименьшего и наибольшего значений функции на отрезке;

определение значения функции в точке;

создание дерева;

удаление дерева.

2.2 Разработка схем алгоритмов программы

Разрабатываемые схемы алгоритмов должны обеспечить создание исходной программы (рисунки 1-5).



Рисунок 1 «Схема алгоритма основной программы»



Рисунок 2 «Схема алгоритма функции find\_min\_max»



Рисунок 3 «Схема алгоритма функции calculate»



Рисунок 4 «Схема алгоритма функции make\_tree»



Рисунок 5 «Схема алгоритма функции clear\_tree»

2.3 Разработка кода программы

Уточнение базовой структуры процедур и функций, используемых во время выполнения программы, позволило разработать код программы.

Ниже приведен код всей программы.

*#define \_USE\_MATH\_DEFINES*

*#include <iostream>*

*#include <sstream>*

*#include <cmath>*

Продолжение кода программы

*using namespace std;*

*enum Type { PLUS, MIN, MUL, DIV, SQU, SIN, COS, TG, CTG, LN, LG, VAL };*

*struct Tree*

*{*

*Type type;*

*Tree\* left, \* right;*

*string data;*

*};*

*void find\_min\_max(Tree\* tree, string s);*

*double calculate(Tree\* tree, double val);*

*void make\_tree(string str, Tree\* node, size\_t len);*

*void clear\_tree(Tree\* tree);*

*int main()*

*{*

*setlocale(LC\_ALL, "Rus");*

*string s("None");*

*Tree\* tree = nullptr;*

*while (true)*

*{*

*cout << "Выберите команду: " << endl;*

*cout << "0 - выход" << endl;*

*cout << "1 - ввод функции" << endl;*

*cout << "2 - нахождение экстремумов функции на отрезке" << endl;*

*cout << "3 - вывод текущей функции" << endl;*

*cout << "\nВаш выбор: ";*

*char ans;*

*cin >> ans;*

*cout << endl;*

*cin.ignore(2, '\n');*

*switch (ans)*

*{*

*case '0':*

*clear\_tree(tree);*

*return 0;*

*break;*

*case '1':*

*clear\_tree(tree);*

*cout << "Введите функцию y = ";*

*getline(cin, s);*

*tree = new Tree;*

*make\_tree(s, tree, s.length());*

*break;*

*case '2':*

*if (tree != nullptr)*

*find\_min\_max(tree, s);*

*else*

*cout << "Пожалуйста, введите функцию" << endl;*

*break;*

*case '3':*

*if (s != "None")*

*cout << "Текущая функция y = " << s << endl;*

Продолжение кода программы

*else*

*cout << "Нет введенной функции" << endl;*

*break;*

*default:*

*cout << "Команда " << ans << " неизвестна" << endl;*

*}*

*cout << endl;*

*system("pause");*

*system("cls");*

*}*

*return 0;*

*}*

*void find\_min\_max(Tree\* tree, string s)*

*{*

*double a, b, n = 1e-6;*

*string a\_string, b\_string;*

*bool key = false;*

*cout << "Введите концы отрезка [a; b]: ";*

*cin >> a\_string >> b\_string;*

*for (int i = 0; i < a\_string.length(); i++)*

*{*

*if (!isdigit(a\_string[i]))*

*{*

*key = true;*

*break;*

*}*

*}*

*for (int i = 0; i < b\_string.length(); i++)*

*{*

*if (!isdigit(b\_string[i]))*

*{*

*key = true;*

*break;*

*}*

*}*

*if (key)*

*{*

*cerr << "Введены неверные значения" << endl;*

*return;*

*}*

*a = atoi(a\_string.c\_str());*

*b = atoi(b\_string.c\_str());*

*double h = (b - a) \* 1.0 / n, min, max, min\_x = a, max\_x = a;*

*min = max = calculate(tree, a);*

*for (double x = a; x <= b; x += n)*

*{*

*double c = calculate(tree, x);*

*if (c > max) { max = c; max\_x = x; }*

*if (c < min) { min = c; min\_x = x; }*

*}*

*double c = calculate(tree, b);*

Продолжение кода программы

*if (c > max) { max = c; max\_x = b; }*

*if (c < min) { min = c; min\_x = b; }*

*cout << "\nФункция y = " << s << endl;*

*cout << "На отрезке [" << a << "; " << b << "]: " << endl << "min = y(" << min\_x << ") = " << min << "; max = y(" << max\_x << ") = " << max << endl;*

*}*

*double calculate(Tree\* tree, double val)*

*{*

*double res = 0;*

*switch (tree->type)*

*{*

*case PLUS:*

*res = calculate(tree->left, val) + calculate(tree->right, val);*

*break;*

*case MIN:*

*res = calculate(tree->left, val) - calculate(tree->right, val);*

*break;*

*case MUL:*

*res = calculate(tree->left, val) \* calculate(tree->right, val);*

*break;*

*case DIV:*

*if (calculate(tree->right, val) != 0)*

*res = calculate(tree->left, val) / calculate(tree->right, val);*

*else*

*{*

*cerr << "Деление на 0" << endl;*

*exit(1);*

*}*

*break;*

*case SQU:*

*res = pow(calculate(tree->left, val), calculate(tree->right, val));*

*break;*

*case SIN:*

*res = sin(calculate(tree->left, val));*

*break;*

*case COS:*

*res = cos(calculate(tree->left, val));*

*break;*

*case TG:*

*res = tan(calculate(tree->left, val));*

*break;*

*case CTG:*

*res = 1.0 / (tan(calculate(tree->left, val)));*

*break;*

*case LN:*

*if (calculate(tree->left, val) > 0)*

*res = log(calculate(tree->left, val));*

*else*

*{*

*cerr << "Аргумент ln должен быть > 0" << endl;*

*exit(1);*

Продолжение кода программы

*}*

*break;*

*case LG:*

*if (calculate(tree->left, val) > 0)*

*res = log10(calculate(tree->left, val));*

*else*

*{*

*cerr << "Аргумент lg должен быть > 0" << endl;*

*exit(1);*

*}*

*break;*

*case VAL:*

*if (tree->data == "x")*

*res = val;*

*else if (tree->data == "pi")*

*res = M\_PI;*

*else if (tree->data == "e")*

*res = M\_E;*

*else*

*res = atoi(tree->data.c\_str());*

*break;*

*default:*

*cerr << "Невозможно посчитать значение в функции в точке " << val << endl;*

*exit(1);*

*}*

*return res;*

*}*

*void make\_tree(string s, Tree\* tree, size\_t len)*

*{*

*unsigned i, c;*

*for (unsigned t = PLUS; t < VAL; ++t)*

*for (i = 0, c = 0; i < s.size(); ++i)*

*{*

*if ('(' == s[i])*

*++c;*

*else if (')' == s[i])*

*--c;*

*else if ((t == PLUS ? '+' : t == MIN ? '-' : t == DIV ? '/' : '\*') == s[i] && !c)*

*{*

*tree->left = new Tree;*

*tree->right = new Tree;*

*tree->type = static\_cast<Type>(t);*

*make\_tree(s.substr(0, i), tree->left, len);*

*make\_tree(s.substr(i + 1, s.size() - i), tree->right, len);*

*return;*

*}*

*else if (t == SQU && s[i] == '^' && !c)*

*{*

*unsigned j, c1;*

*tree->left = new Tree;*

Продолжение кода программы

*tree->right = new Tree;*

*tree->type = static\_cast<Type>(t);*

*make\_tree(s.substr(0, i), tree->left, len);*

*for (j = i + 1, c1 = 0; j < s.size() && (c1 || isdigit(s[j]) || s[j] == '('); ++j)*

*{*

*if ('(' == s[j])*

*++c1;*

*else if (')' == s[j])*

*--c1;*

*}*

*make\_tree(s.substr(i + 1, j - i - 1), tree->right, len);*

*return;*

*}*

*else if (s.size() - i >= 3)*

*{*

*if (((t == SIN ? "sin" : t == COS ? "cos" : "...") == s.substr(i, 3) ||*

*(s.substr(i, 3) == "ctg" && t == CTG) || (s.substr(i, 2) == "lg" && t == LG)*

*|| (t == TG ? "tg" : t == LN ? "ln" : "..") == s.substr(i, 2)) && !c)*

*{*

*tree->left = new Tree;*

*tree->right = nullptr;*

*tree->type = static\_cast<Type>(t);*

*if (t == SIN || t == COS || t == CTG)*

*{*

*unsigned j, c1;*

*for (j = i + 3, c1 = 0; j < s.size() && (c1 || s[j] == '('); ++j)*

*{*

*if ('(' == s[j])*

*++c1;*

*else if (')' == s[j])*

*--c1;*

*}*

*make\_tree(s.substr(i + 3, j - i - 1), tree->left, len);*

*}*

*else*

*{*

*unsigned j, c1;*

*for (j = i + 2, c1 = 1; j < s.size() && c1; ++j)*

*{*

*if ('(' == s[j])*

*++c1;*

*else if (')' == s[j])*

*--c1;*

*}*

*make\_tree(s.substr(i + 2, j - i - 1), tree->left, len);*

*}*

*return;*

Продолжение кода программы

*}*

*}*

*}*

*if (s[0] == '(' && s[s.size() - 1] == ')')*

*{*

*make\_tree(s.substr(1, s.size() - 2), tree, len);*

*return;*

*}*

*if (s.length() == len && s != "x")*

*{*

*cerr << "\nНеизвестная функция y = " << s << endl;*

*exit(1);*

*}*

*tree->type = VAL;*

*stringstream istr(stringstream::in | stringstream::out);*

*istr << s;*

*istr >> tree->data;*

*tree->left = nullptr;*

*tree->right = nullptr;*

*}*

*void clear\_tree(Tree\* tree)*

*{*

*if (!tree) return;*

*if (tree->left) clear\_tree(tree->left);*

*if (tree->right) clear\_tree(tree->right);*

*delete tree;*

*}*

2.4 Тестирование программы

Разработанное консольное приложение позволяет выполнять функции, предусмотренные заданием (рисунки 6-9) .

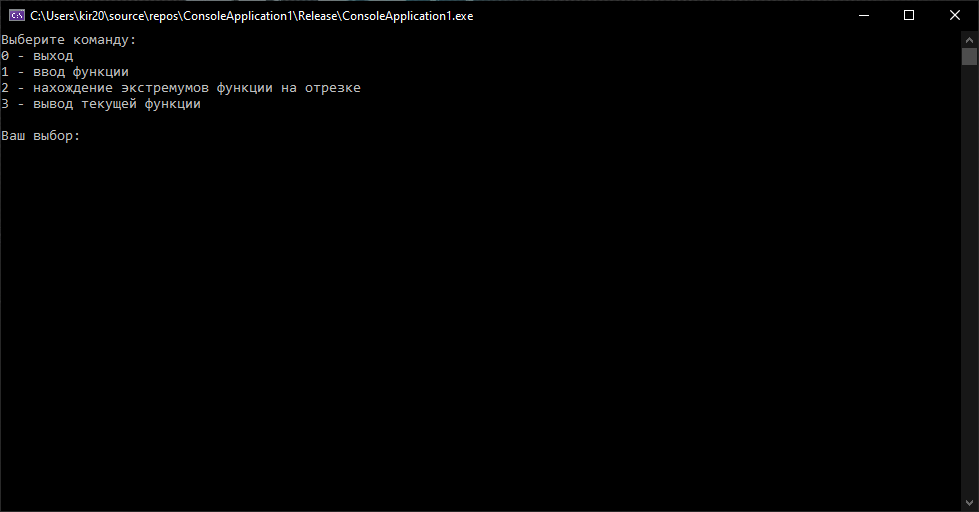


Рисунок 6 «Меню»

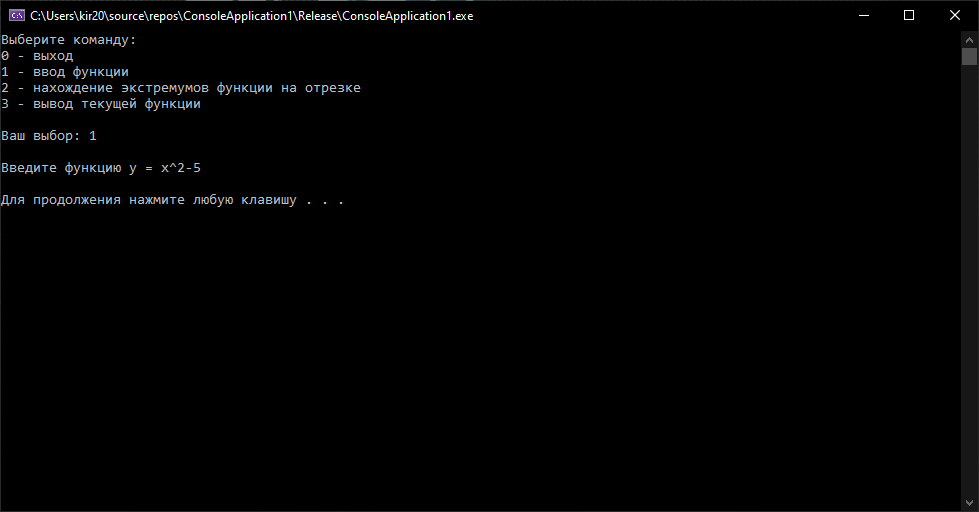


Рисунок 7 «Добавление функции»

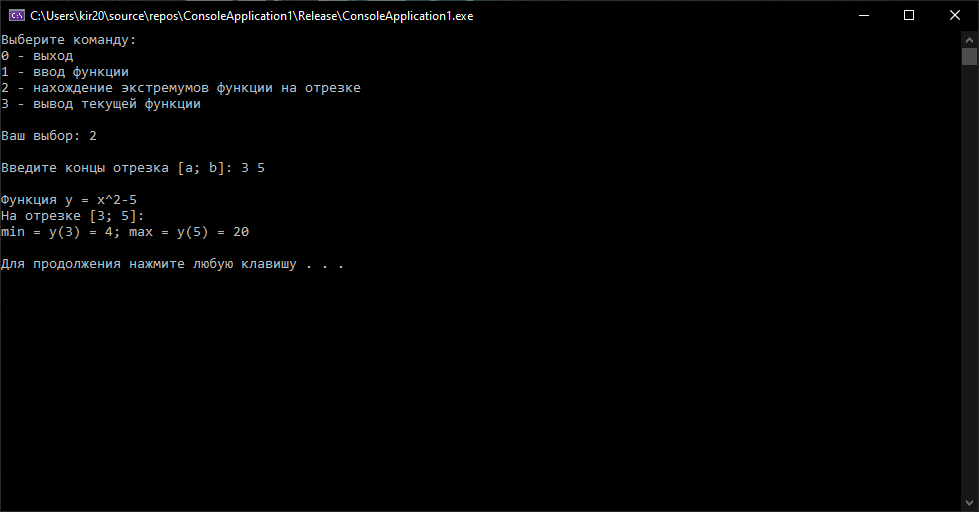


Рисунок 8 «Поиск экстремумов функции на отрезке»

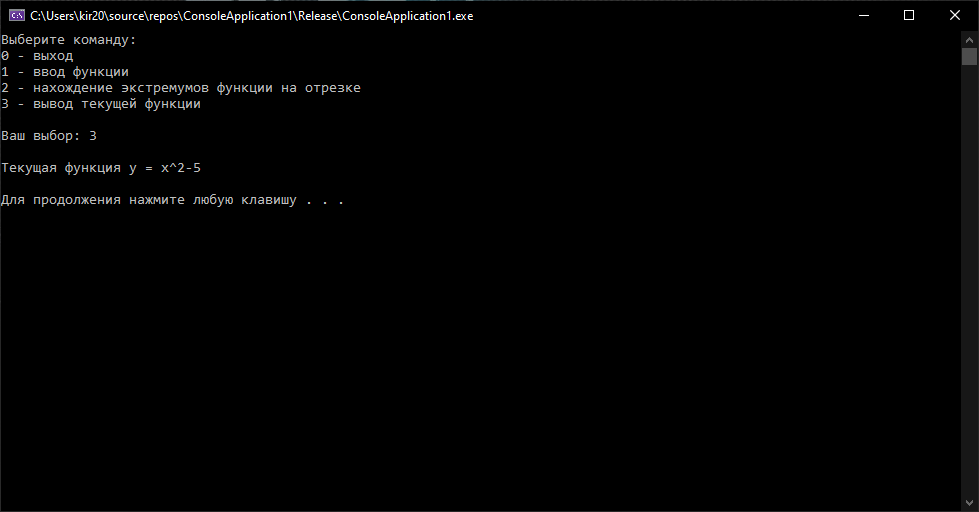


Рисунок 9 «Вывод информации о текущей функции»

Вывод

При разработке программы были изучены средства разработки программ с элементарным интерфейсом консольного режима на языке C++, используемые при структурном подходе, а также схемы алгоритмов, сопровождающие процесс разработки.

# 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).

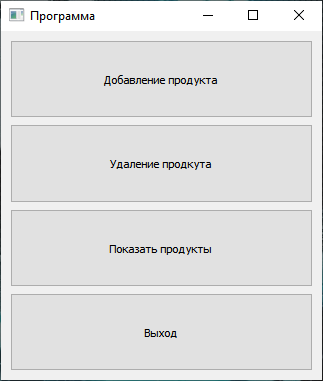


Рисунок 2 – Внешний вид формы «Программа»

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

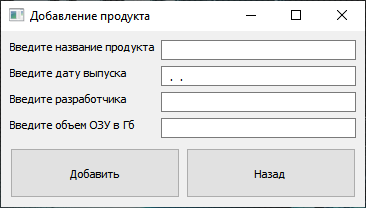


Рисунок 3 – Внешний вид формы «Добавление данных»

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

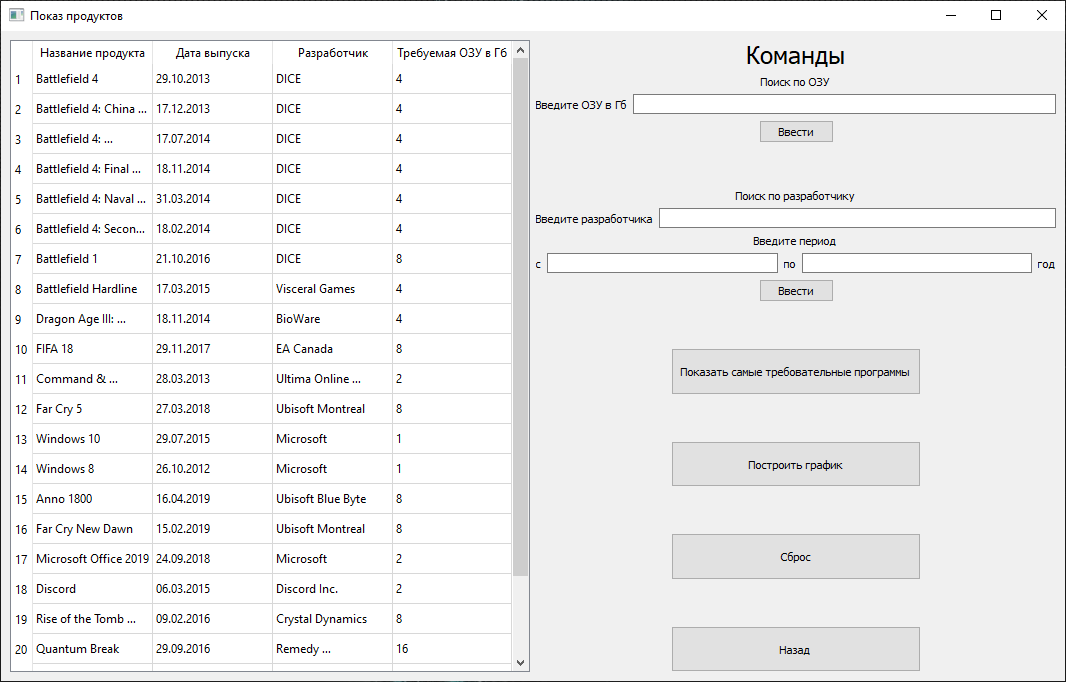


Рисунок 4 – Внешний вид формы «Показать продукты»

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 исходного 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) .

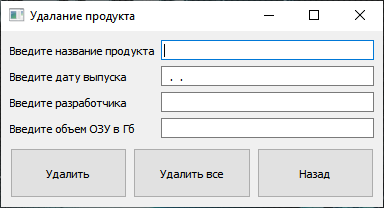


Рисунок 9 «Окно удаления элементов»

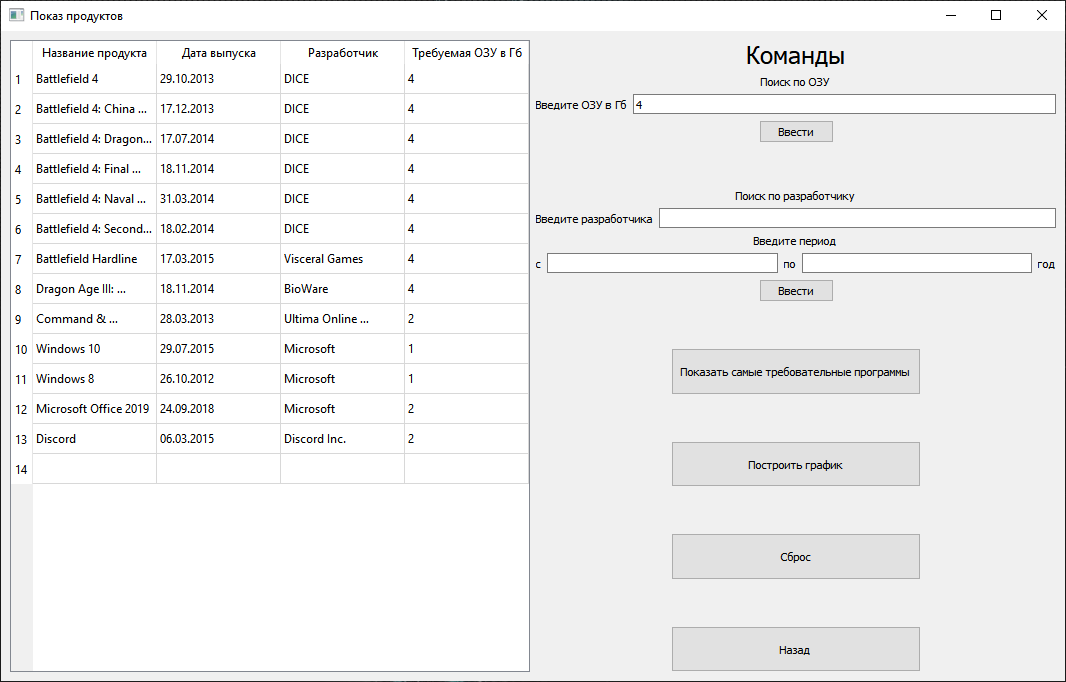


Рисунок 10 «Окно выполнения поиска продуктов по ОЗУ»

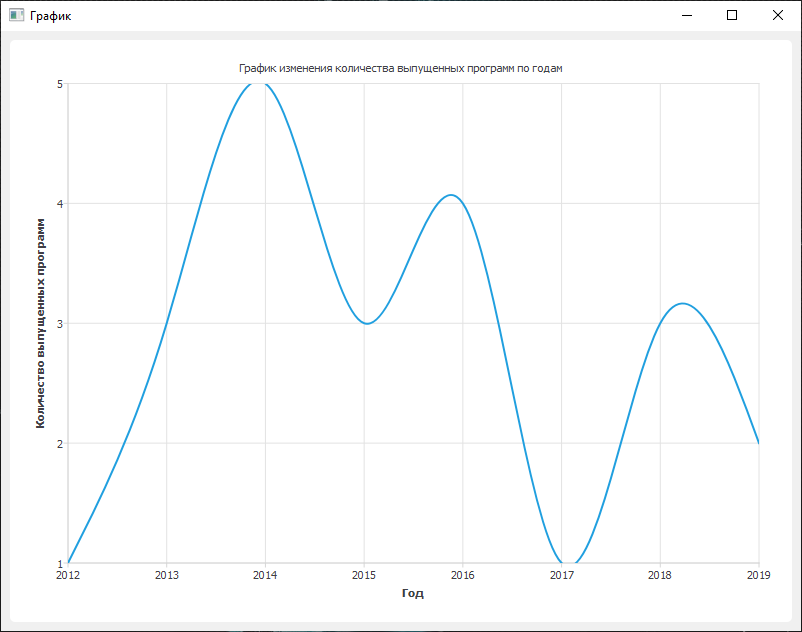


Рисунок 11 «Окно график изменения количества выпущенных программ по годам»

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.

# Заключение

Во время выполнения учебной практики более глубоко были изучены средства реализации проектов программ на Object Pascal и С++, получены практические навыки проектирования небольших программных систем при структурном и объектном подходах, с консольным и графическими интерфейсами.

Кроме этого были получены начальные навыки составления программной документации.

# Список литературы

1. Иванова Г.С., Ничушкина Т.Н. Объектно-ориентированное программирование: учебник. – М.: Издательство МГТУ им. Н.Э. Баумана, 2014.

2. Иванова Г.С. Программирование. Учеб. для вузов – М.: Кнорус, 2017.

3. Иванова Г.С., Ничушкина Т.Н. Консольные приложения С++ в среде Microsoft Studio 2008 (Visual C++): Методические указания по выполнению лабораторных работ. – М.: МГТУ им. Н.Э. Баумана, 2013. (<http://e-learning.bmstu.ru/moodle/course/view.php?id=129>).

4. Самарев Р.С. Программирование с использованием библиотеки Qt: Методические указания по выполнению лабораторной работы. М.: Издательство МГТУ им. Н.Э. Баумана, 2011. (<http://e-learning.bmstu.ru/moodle/course/view.php?id=129>).

5. Шлее М. Qt4. 5. Профессиональное программирование на C++ / Шлее М. - СПб. : БХВ-Петербург, 2018. - 884 с. + CD. - (В подлиннике). - ISBN 978-5-9775-0398-3.

6. Шилдт Г. Полный справочник по С++, 4 изд.М.: Изд. дом «Вильямс», 2015.