**курсовой проект**

**МДК 01.02 «Прикладное программирование»**

(наименование дисциплины)

Министерство образования, науки и молодежной политики  
Краснодарского края

Государственное БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ образовательное учреждение

КРАСНОДАРСКОГО КРАЯ

«Новороссийский колледж радиоэлектронного приборостроения»

Приложение для контроля времени работы с процессом

тема курсового проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Пояснительная записка**

Оценка за пояснительную записку \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка за графическую часть \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка за защиту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка общая \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Выполнила**

**Студент** \_\_\_\_\_\_\_3-П-1\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кравчина А.А. \_\_\_\_\_\_\_\_\_

(номер и шифр группы) (подпись) (фамилия, имя, отчество)

**Руководитель** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шепилова А.А.\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, имя, отчество)

Содержание

[Введение 5](#_Toc511990474)

[1 Постановка задачи 6](#_Toc511990475)

[2 Исследование задачи 7](#_Toc511990476)

[3 Реализация задачи 8](#_Toc511990477)

[Список использованных источников 23](#_Toc511990478)

# Введение

Где расположить диван, а куда поставить кровать, а хватит ли места?

С помощью данной программы можно не просто узнать, а увидеть, всю картину вашей квартиры с мебелью расположенной пол вашему вкусу.

Если заниматься расстановкой мебели в своей квартире, то наличие возможности взглянуть на всю комнату с мебелью просто необходимо. Программа позволит планировать расположение всей мебели в комнате. Вы сможете не беспокоиться об отсутствии или нехватке места для мебели или комфортного проживания.

При не правильном планировании или его отсутствии, очень велика вероятность ошибиться в расстановке мебели, и в итоге вы можете остаться недовольным или вовсе потерять деньги из-за не хватки места для размещения нового шкафа.

Данный курсовой проект выполнен на языке высокого уровня C++ с использованием среды разработки Embarcadero RAD Studio XE.

Целью курсового проекта является создание «Приложения для расстановки мебели» в виде исполняемого файла программы.

# Постановка задачи

Для выполнения данного курсового проекта, была поставлена цель разработать «Приложение для расстановки мебели» в виде исполняемого файла программы со следующими функциями:

1. На форме отображаются следующие элементы:
   * + Главная панель на которой отображается комната;
     + верхнее меню, состоящее из стандартных элементов: «Стол», «Кровать», «Диван»;
     + панель управления выбранным элементом;
     + список всех элементов;
     + при нажатии в меню на элемент появляется форма для ввода имени, а после добавляется в список элементов;
     + при нажатии на элемент в списке, им можно манипулировать;
     + при двойном нажатии на элемент в списке открывается форма форматирования элемента, где можно изменить размеры, цвет и имя.
2. Исследование задачи

Для реализации «Приложение для расстановки мебели» понадобится:

Среда разработки - комплекс программных средств, используемый программистами для разработки программного обеспечения:

* Компилятор (если отсутствует в среде разработки);

Для курсового проекта была выбрана – Embarcadero RAD Studio XE среда быстрой разработки приложений (RAD) фирмы Embarcadero Technologies, работающая под Windows. Несмотря на то, у Embarcadero RAD Studio XE проприетарная лицензия, она упрощает задачу разработки приложений, поскольку имеет программируемые модули.В состав Embarcadero RAD Studio XE входит Embarcadero C++ Builder — это среда C++, которая полностью соответствует концепции быстрой разработки приложений, объединяет средства ANSIC++ и многофункциональную расширяемую инфраструктуру визуальных компонентов.

Следующий шаг – изучение алгоритма работы «Приложения для расстановки мебели», в котором пользователь, нажимая на один из пунктов меню «Мебель», создаст новый объект. Воспользовавшись элементами управления, может изменить место положения объекта на панели отображения. На этой форме расположена кнопка «Удалить», которая удаляет текущий объект. При двойном нажатии на элемент откроется окно его форматирования, где можно изменить его цвет, размер и имя.

Проведя исследование задачи, можно приступить к реализации приложения.

# Реализация задачи

Для начала, необходимо создать основные формы: Главная форма, форма форматирования элемента.

На главной форме располагаются следующие элементы: MainMenu, Panel1, Panel2, Panel3, Image1, Button(7), ListBox.

В MainMenu находиться элемент «Мебель», нажав на который появляется выпадающий список со стандартными элементами «Стол», «Кровать», «Кресло».

Элемент Panel1 содержит элемент Image1, который рисует все элементы, который создал пользователь. В Panel2 находятся элементы Label, Edit, Button(6), которые отвечают за идентификацию активного элемента и его перемещения. Элемент Panel3 содержит компоненты Button и ListBox, отвечающие за вызов окна форматирования элемента, удаления элементов, и предоставление информации о созданных элементах пользователя.

Интерфейс первой формы показан на рисунке (рисунок 1):

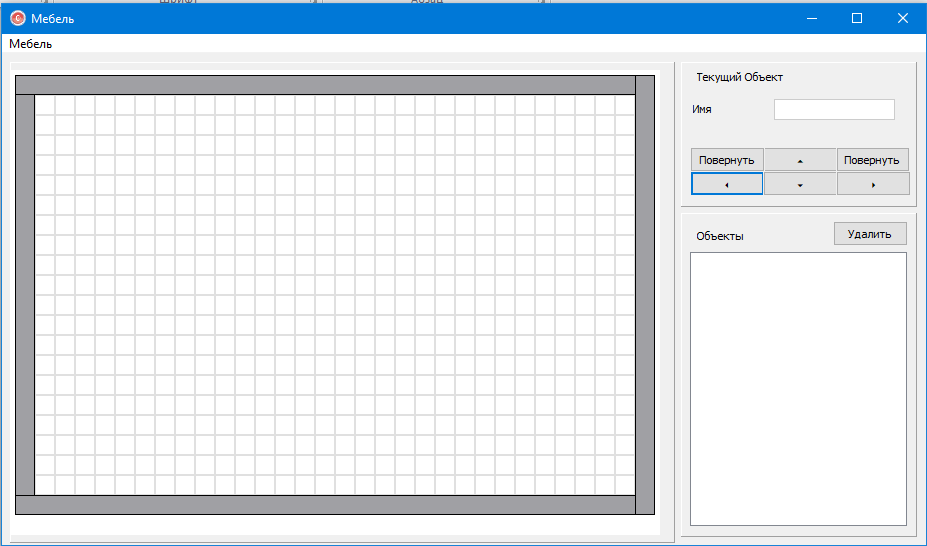


Рисунок 1 – Интерфейс первого окна

На форме для форматирования элементов мебели расположены компоненты: Panel1, Panel2, Image1, Edit, Label, Button(2).

На компоненте Panel1 находиться компонент Image1, который рисует выбранный элемент мебели с принимаемыми изменениями. В компоненте Panel2 находятся компоненты для ввода и изменения свойств компонента, такие как Edit и компоненты Panel которые имитируют кнопки для изменения цвета.

Интерфейс формы форматирования элементов показан на рисунке (рисунок 2):

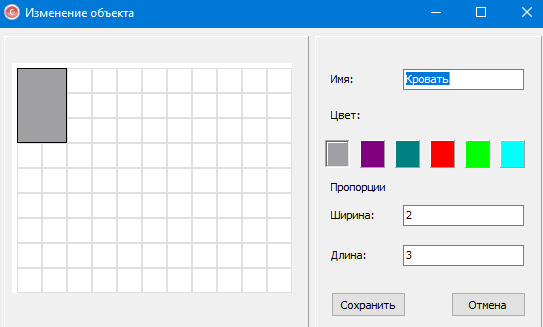


Рисунок 2 – Интерфейс окна форматирования элемента

Для начала, необходимо создать новую форму для будущей программы, разместить на ней необходимые компоненты: Label,CheckListBox, MainMenu и подключить необходимую библиотеку #include <string>:

#include <vcl.h>

#pragma hdrstop

#include <string>

#include "Unit1.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

Директива **#include <vcl.h>** подключает заголовочный файл **vcl.h**, содержащий объявления, используемые в библиотеке визуальных компонентов C++Builder.

Директива **#pragma hdrstop** указывает компилятору конец списка общих файлов.#include <string>-это класс с методами и переменными для организации работы со строками в языке программирования C++.

#include "Unit1.h"

Следующая директива включает файл **Unit1.h,** который ищется, прежде всего, в каталоге, в котором расположен файл, содержащий данную директиву.  
#pragma package(smart\_init)

#pragmaresource"\*.dfm"

Первая из них определяет последовательность инициализации пакетов такой, какая устанавливается взаимными ссылками использующих их модулей. Вторая говорит препроцессору, что для формы надо использовать файл **.dfm** с тем же именем, что и имя данного файла.

TForm1\*Form1;  
Так объявляется указатель с именем Form1 на объект класса TForm1, в дальнейшем он может использоваться для доступа к свойствам и методам формы с именем "Form1".

Зададим иконку программы в свойстве Icon.Зададим цвет clSkyBlue формы в свойстве Color и имя формы в свойстве Caption:Timer. Также обозначим в компоненте MainMenu пункты меню: «Начало», «Запуск», «Очистка», «Поиск» , «Выход».

Необходим запуск для проверки работоспособности программы (рисунок 1):

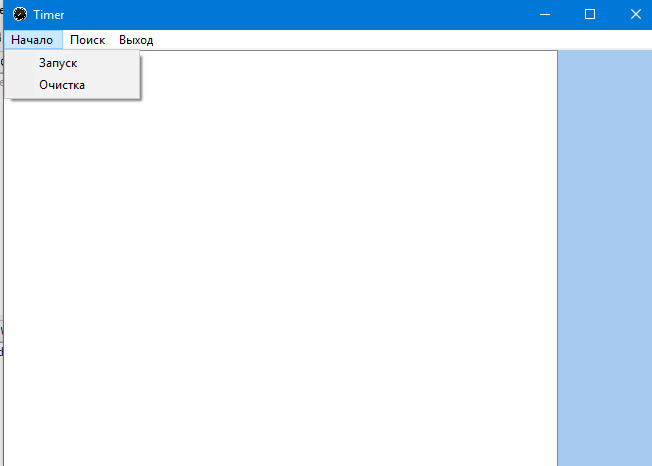


Рисунок 1 – Результат работы программы

Как видно, программа запускается без проблем.

Теперь можно приступить к реализации функции, которая будет выводить список активных процессов в CheckListBox, при нажатии на кнопку «Запуск»:

HWND h = Handle;

//задает переменную h как дескриптор окна

wchar\_t buf[128];

//задает массив строк(буфер), в котором будет хранится дескриптор каждого следующего окна

CheckListBox1->Clear();

//очищает СheckListBox1

CheckListBox1->Items->Clear();

//очищаем массив объектов CheckListBox1

do {

h = GetNextWindow(h, GW\_HWNDNEXT);

//присваивает переменной h значение следующего окна

GetWindowText(h, buf, 128);

//копирует текст строки заголовка определяемого окна h в buf

if (buf[0]) CheckListBox1->Items->Add(buf);

//обращаемся к 1 элементу массива buf и добавляем его в CheckListBox1

} while (h != NULL);

//пока h не пустой выполняется цикл do-while

CheckListBox1->ItemIndex = 0;

//задаем номер выбранного элемента равным 0

CheckListBox1->Update();

//происходит обновление CheckListBox1

На рисунке 2 показан результат работы этой кнопки:

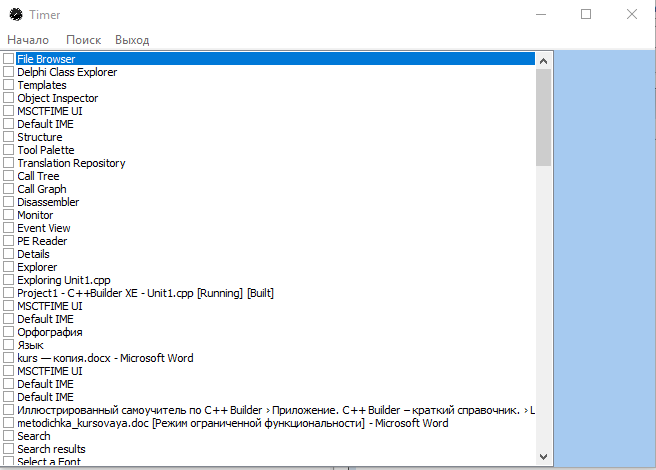


Рисунок 2 – Проверка работоспособности программы после изменения

Теперь в CheckListBox1 отображается список процессов, существующих в данный момент в системе. Для клавиши «Очистка» задаем следующее:

CheckListBox1->Items->Clear();

//происходит очистка массива обьектов Items

Для клавиши «Выход»:

Form1->Close();

//закрывает Form1

В событии OnResize задаем:

Form1->Refresh();

//обновление содержания формы

Далее необходимо реализовать функцию поиска необходимого процесса по названию. Для начала создаем новую форму и добавляем ее к заголовку Form1 таким образом: #include "Unit2.h". Зададим цвет clSkyBlue Form2 в свойстве Color. Зададим иконку Form2 в свойстве Icon. В Form2 добавляем таким же образом:#include "Unit1.h",меняем в свойстве Caption название на «Поиск», добавляем компоненты: Button1, Edit1, Label1, Image1.

В Label1 изменяем Caption на «Введите слово»,в Edit1 очищаем свойство Text,в Button1 меняем Caption на «Ок»,в Image1 в свойстве Picture загружаем картинку формата .png.

Подключаем библиотеку #include <vector> для использования вектора

Объявляем глобальные переменные:

std::vector<HWND> originDesc;

//динамический массив дескрипторов

Cоздаем функцию void getDescriptors(), в теле которой мы осуществляем получение дескрипторов всех процессов, работающих в системе:

originDesc.clear();

//очищает динамический массив дескрипторов

HWND h = Form1->Handle;

//задает дескриптор окна с Form1

do {

h = GetNextWindow(h, GW\_HWNDNEXT);

// присваивает переменной h значение следующего окна

originDesc.push\_back(h);

//заносит в массив значения окон

} while (h != NULL);

//пока h не пустой выполняется цикл do-while

Теперь осуществим функцию descksPrint(),которая будет осуществлять непосредственно поиск и вывод нужного процесса:

Form1->CheckListBox1->Clear();

//очистка CheckListBox1

std::string search = AnsiString(Form2->Edit1->Text).c\_str();

/\*вводим строковую переменную search и присваиваем ей значение из свойства Text компонента Edit1.

Используя функцию c\_str(),которая возвращает указатель на символьный массив, для того, чтобы перевести строку из одного типа данных в другой(в AnsiString) \*/

wchar\_t buf[128];

//задает массив строк(буфер), в котором будет хранится дескриптор каждого следующего окна

std::vector<HWND>::iterator it;

//осуществляется инициализация итератора при помощи vector

for (it=originDesc.begin();it!=originDesc.end();it++){

//цикл for по массиву дескрипторов

GetWindowText(\*it, buf, 128);

//получение заголовка окна дескриптора

if (buf[0]){

//если 1 элемент массива buf cуществует

if(search.length()!=0){

//если длина строки search не равняется 0

std::wstring ws(buf);

//объявление строки ws типа wstring cодержащей buf

Это нужно для того, чтобы разобрать содержимое этой строки.

std::string name(ws.begin(), ws.end());

//преобразует строку wstring в string

if(name.find(search) != std::string::npos)

//если вхождение в строку name строкой search найдено

Form1->CheckListBox1->Items->Add(UnicodeString(buf));

//то добавляет строку в массив Items компонента CheckListBox1

}else{

Form1->CheckListBox1->Items->Add(UnicodeString(buf));

// иначе,добавляет строку в массив Items компонента CheckListBox1

В событие Edit1Change компонента Edit1 добавим функцию descksPrint() для поиска названия процесса при вводе названия. В Button1Click компонента Button1 добавим функции getDescriptors(),которая выводит список процессов, и descksPrint(),осуществляющая поиск названия процесса. Для компонента Image добавим событие Image1Click и пропишем в нем Form2->Edit1->Clear() для очистки поля ввода названия процесса.

На рисунке 3 проверка функций:

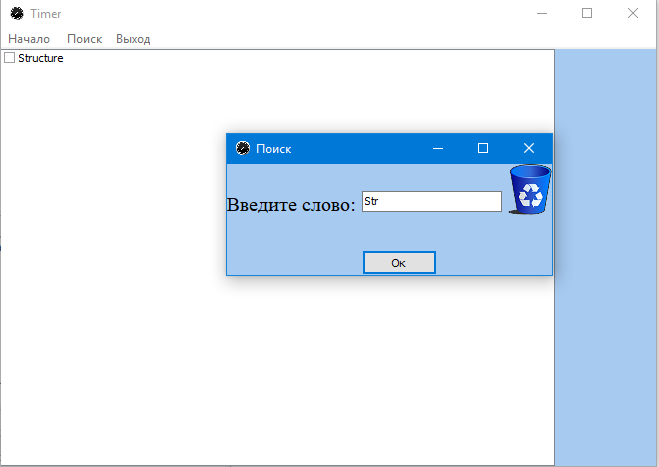


Рисунок 3–Функция поиска

Для реализации самого таймера потребуется создать Form3,разместить в ней компоненты Label1,Label2,Label3,Button1,Timer1,а также поменять цвет Form3 в свойстве Color на clSkyBlue.Зададим иконку Form3 в свойстве Icon.В свойстве Caption нужно поменять название на Timer.В заголовке Form3 подключаем предыдущие формы таким образом:

#include "Unit2.h"

#include "Unit1.h"

Так же в Form1 и Form2 подключаем Form3:

#include "Unit3.h"

Подключаем библиотеку #include <string> для выполнения действие со строками. Теперь нужно обозначить глобальные переменные:

UnicodeString name;

unsigned int tm = 0;

bool stop = false;

Теперь нужно описать функцию void checkProcess(),которая проверяет существует ли процесс:

if(name.Length()==0) return;

//если длина строки равна 0 то возврат в начало вызова функции

HWND hwnd;

//объявление дескриптора окна

hwnd = FindWindowW(NULL, name.c\_str());

//дескриптор равен функции FindWindowW,которая получает дескриптор окна с именем name

if (hwnd != 0) {

} else{

stop=true;

Form3->Close();

}

//если дескриптор равен 0,то присваивается переменной stop значение true и закрывается Form3

Необходимо описать функцию setProcessName(UnicodeString p\_name),которая производит получение имени процесса:

name = p\_name;

//переприсвоение имени получаемого процесса строке name

Сейчас нужно ввести функцию void drawtime(),которая осуществляет сам таймер:

int seconds = tm%60;

int hours = tm/3600;

int minutes = tm/60%60;

UnicodeString Shours = (hours<10)?"0"+IntToStr(hours):IntToStr(hours);

//если переменная hours меньше 10,переводит hours из целого числа в строку

и присваивает переменной Shours

UnicodeString Sminutes = (minutes<10) ?"0" +IntToStr (minutes): IntToStr (minutes);

//если переменная minutes меньше 10,переводит minutes из целого числа в строку и присваивает переменной Sminutes

UnicodeString Sseconds= (seconds<10) ?"0"+ IntToStr (seconds): IntToStr (seconds);

//если переменная seconds меньше 10,переводит minutes из целого числа в строку и присваивает перменной Sseconds

Form3->Label1->Caption = Shours+":"+Sminutes+":"+Sseconds;

//в свойстве Caption компонента Label1 отображается время работы процесса

В событии OnTimer компонента Timer1:

if(!stop) tm++, checkProcess(), drawtime();

//если переменная stop не false,то увеличивается переменная tm,и включаются функции checkProcess(), drawtime()

В событии OnClick компонента Button 1:

Button1->Caption = (! stop)?"Продолжить":"Стоп";

//если stop равняется true,то в свойстве Caption компонента Button1 пишется «Продолжить»,если false то «Стоп»

Label1->Font->Color = (! stop)?clRed:clBlack;

//если stop равняется true,то красный цвет шрифта в компоненте Label1,если false то черный цвет шрифта

В событии OnClose обозначается, то, что при закрытии Form1,нажимая на кнопку N3, закрывается и Form3:

tm=0;

stop = true;

Form1->N3->Click();

В событии OnShow при показе Form3 переменные обнуляются:

tm=0;

stop = false;

Необходимо перейти к Form1 и описать в компоненте CheckListBox1 событие OnClickCheck,в котором:

UnicodeString act = CheckListBox1->Items->Strings[CheckListBox1->ItemIndex];

//присвоение строке act индекса одного из объектов

Form3->setProcessName(act);

//получение имени процесса с помощью функции setProcessName

Form3->Show();

//показ формы с таймером

На рисунке 4,5 проверка вывода таймера и работы функций :

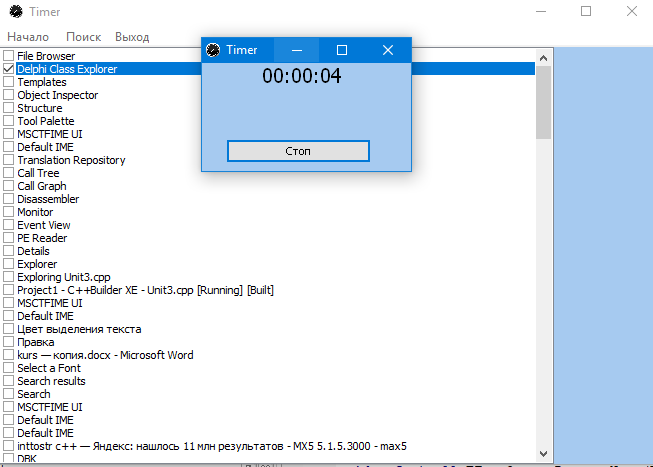


Рисунок 4–Вывод таймера

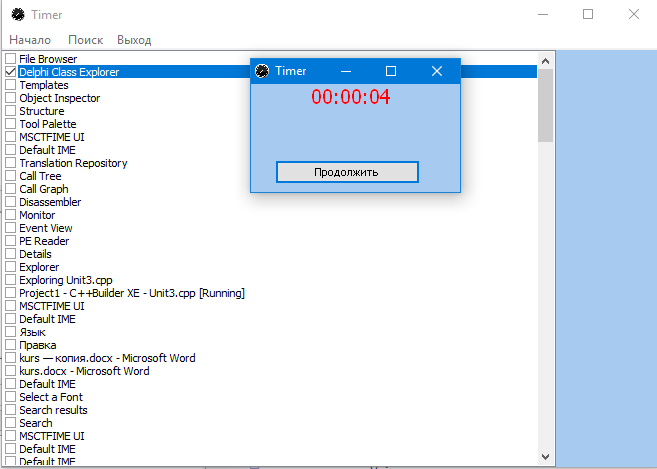


Рисунок 5–Вывод таймера

Теперь можно проверить все функции программы.

1.Вывод процессов (рисунок 6)

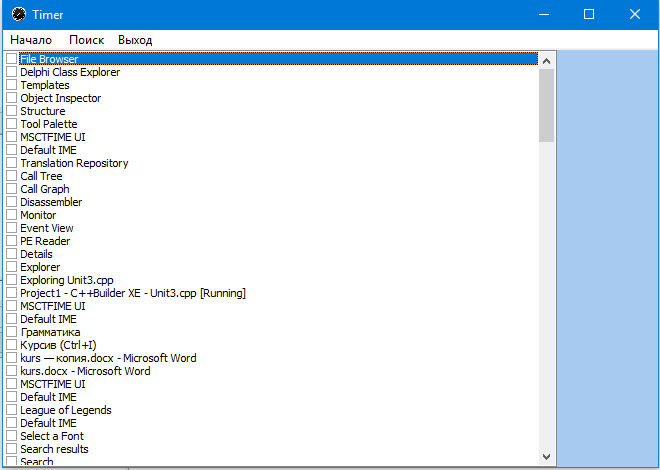


Рисунок 6–вывод процесса

2.Очищение поля с помощью клавиши «Очистка» (рисунок 7)

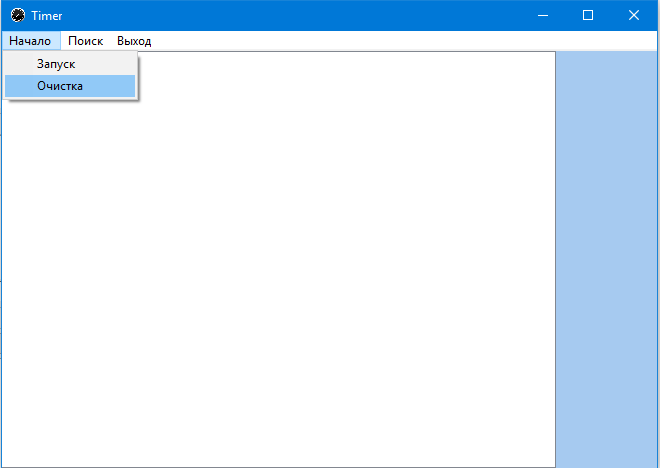


Рисунок 7– очищение поля с помощью клавиши «Очистка»

3.Реализация функции поиска (рисунок 8)

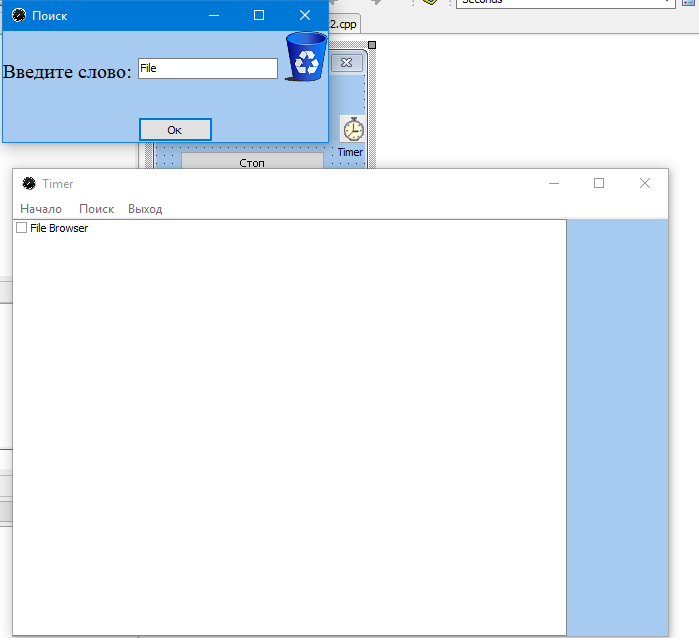


Рисунок 8– реализация функции поиска

4.Очищение поля поиска (рисунок 9)

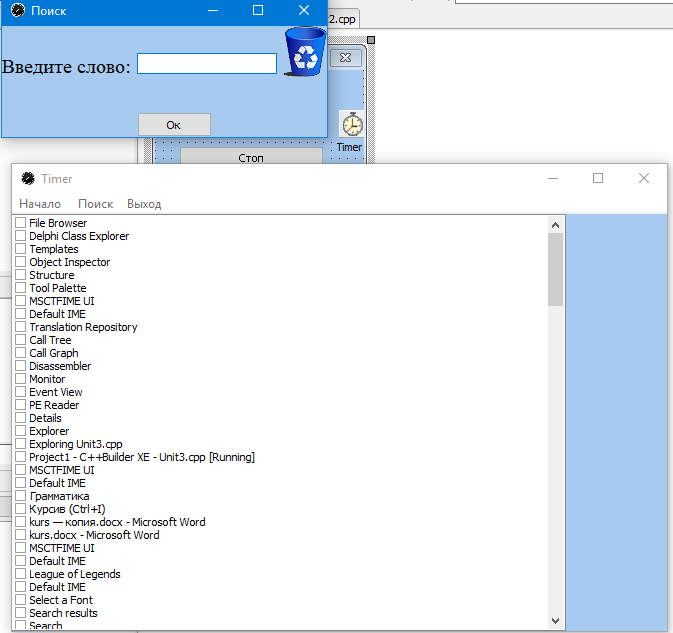


Рисунок 9– очищение поля поиска

5.Вывод таймера по клику (рисунок 10)

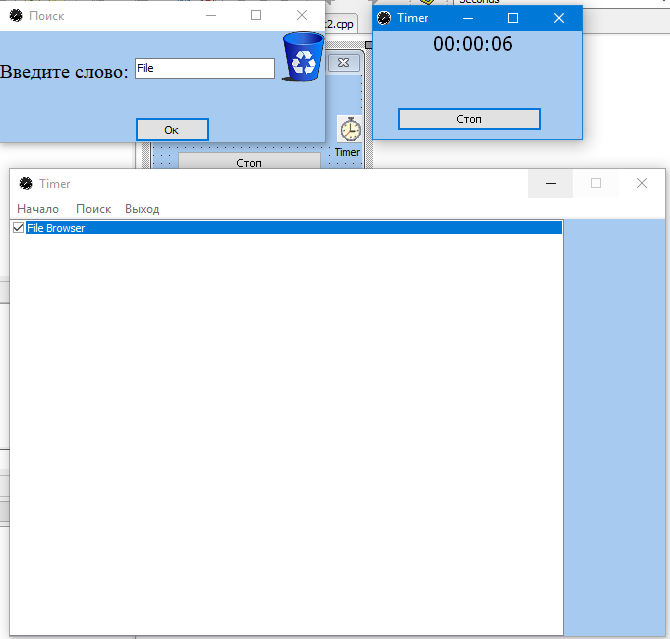


Рисунок 10– вывод таймера по клику

6.Остановка таймера (рисунок 11)

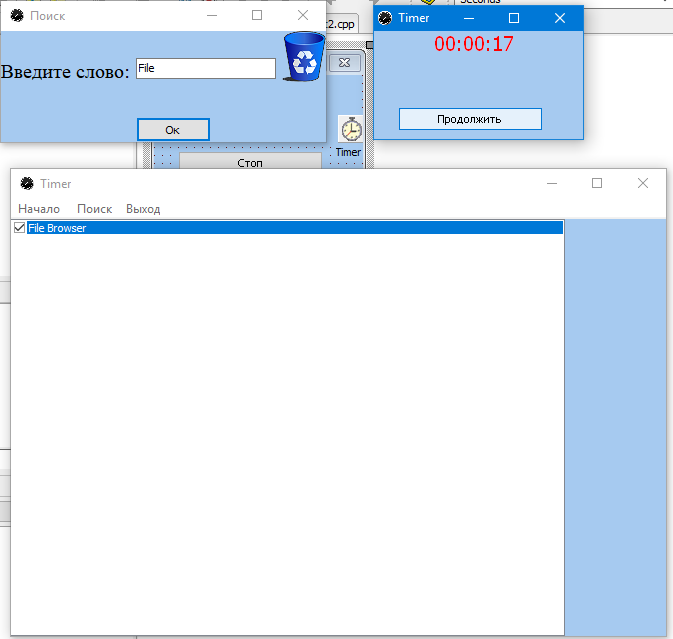


Рисунок 11– остановка таймера

Заключение

В ходе курсового проекта было создано «Приложение для контроля времени работы с процессом» и выполнены следующие задачи:

На форме отображаются следующие элементы:

* + - поле с чекбоксами;
    - верхнее меню, состоящее из элементов: «Начало», «Поиск», «Выход»;
    - сброс таймера;
    - при нажатии на Поиск появляется вторая форма, на которой можно искать нужный процесс;
    - при нажатии на «Начало» есть кнопки: «Очистить», «Запуск». «Очистить» очищает поле с чекбоксами, «Запуск» загружает процессы в поле с чекбоксами;
    - при нажатии на чекбокс нужного процесса, появляется форма с таймером.
    - таймер имеет кнопку «Остановить»

# Список использованных источников

1. Электронный ресурс - Свободная энциклопедия [http://ru.wikipedia.org]
2. Иванова, Г.С. Технология программирования. – М.: КноРус, 2014.
3. Методы оптимизации: Учебное пособие / А.В. Аттетков, В.С. Зарубин, А.Н. Канатников. - М.: ИЦ РИОР: НИЦ Инфра-М, 2013.
4. Х.М.Дейтел, Как программировать на С/С++: Книга 1, Основы программирования. [Текст], Пер. с англ. – М.: ООО «Бином-Пресс», 2013.
5. Х.М.Дейтел, Как программировать на С/С++: Книга 2, Программирование для сети, структуры данных. [Текст] / П.Дж. Дейтел. Пер. с англ. – М.: ООО «Бином-Пресс», 2013.
6. Страуструп, Б. Язык программирования C++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. - М.: БИНОМ, 2012.
7. Ашарина И.В., Крупская Ж.Ф. Язык С++ и объектно-ориентированное программирование в С++. Лабораторный практикум. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2015. – 232 с.
8. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения: учебное пособие / под ред. Л.Г. Гагариной. – М.: ИД "ФОРУМ": ИНФРА-М, 2013. – 400 с.: ил. – (Высшее образование)
9. Лафоре Р. Объектно-ориентированное программирование в С++. Классика Computer Science. 4-е изд. – СПб.: Питер, 2016. – 928 с.
10. Культин Н.Б. C++ Builder в задачах и примерах. – СПб.: БХВ-Петербург, 2012. – 336 с.
11. Немцова Т.И., Голова С.Ю., Терентьев А.И. Программирование на языке высокого уровня. Программирование на языке С++: учебное пособие / под редакцией Л.Г. Гагариной. – М.: ИД "ФОРУМ": ИНФРА-М, 2014. – 512 с.: ил. – (Профессиональное образование)
12. Подбельский В.В. Стандартный С++: учебное пособие / В.В. Подбельский. – М.: Финансы и статистика, 2013. – 688 с.
13. Астахова И.Ф., Власов С.В., Фертиков В.В., Ларин А.В. – Язык С++: Учебное пособие. – мн.: Новое знание, 2012 – 203 с.
14. Бобровский С.И. Технологии С++ Builder. Разработка приложения для бизнеса. Учебный курс. – СПб: Питер, 2013.
15. Васильев А.Н. Самоучитель С++ с примерами и задачами. 3-е издание (переработанное). – СПб.: Наука и Техника, 2015. – 480 с.
16. Джонсон М Харт Системное программирование в среде Windows – 3-е издание Пер. с англ.: – М.: Издательский дом "Вильямс", 2015 г. – 592 с.
17. Орлов С.А. Технологии разработки программного обеспечения: Учебник для вузов. 3-е изд. – СПб.: Питер, 2012. –527 с.: ил.
18. Тимофеев В.В. Самоучитель С++ как он есть. – М.: Издательство Бином, 2014. – 336 с., ил.
19. Эпштейн М.С. Практикум по программированию на языке С, 2007 ОИЦ «Академия».