МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



Кафедра «СМАРТ технологии»

Лабораторная работа № 5:

«Использование графических возможностей приложении на C#»

По дисциплине: «Программирование и алгоритмизация на языках

высокого уровня»

Группа 211-328

Студент Пан Филипп Юльевич

Дата \_\_\_\_\_\_\_\_\_\_\_

Преподаватель Идиатуллов Тимур Тофикович

2022

Задание

L05\_5. Создайте систему климатической установки в технических помещениях

1. Разработать приложение для поддержания температуры в помещениях завода:

* Сбора данных о температуре входного воздуха;
* Расчет мощности работы системы охлаждения в зависимости от потока охлаждаемого воздуха и его начальной температуры;
* Мониторинг температуры в разных помещениях предприятия;
* Контроль за превышением температуры и расчет интенсивности потока охлаждающего воздуха через климатическую установку.

2. Считать все алгоритмы управления работающими по линейному закону. Считать объемы помещений равными 100 куб.м. Принять, что оборудование в помещениях нагревает окружающий воздух на

1-5 градусов (случайно) в течение часа. Положить, что система охлаждения должна обеспечивать

поток воздуха разной интенсивности (0-5 куб. м/мин) с постоянной температурой 20oC.

3. Построить графики работы климатической системы и измеряемой температуры.

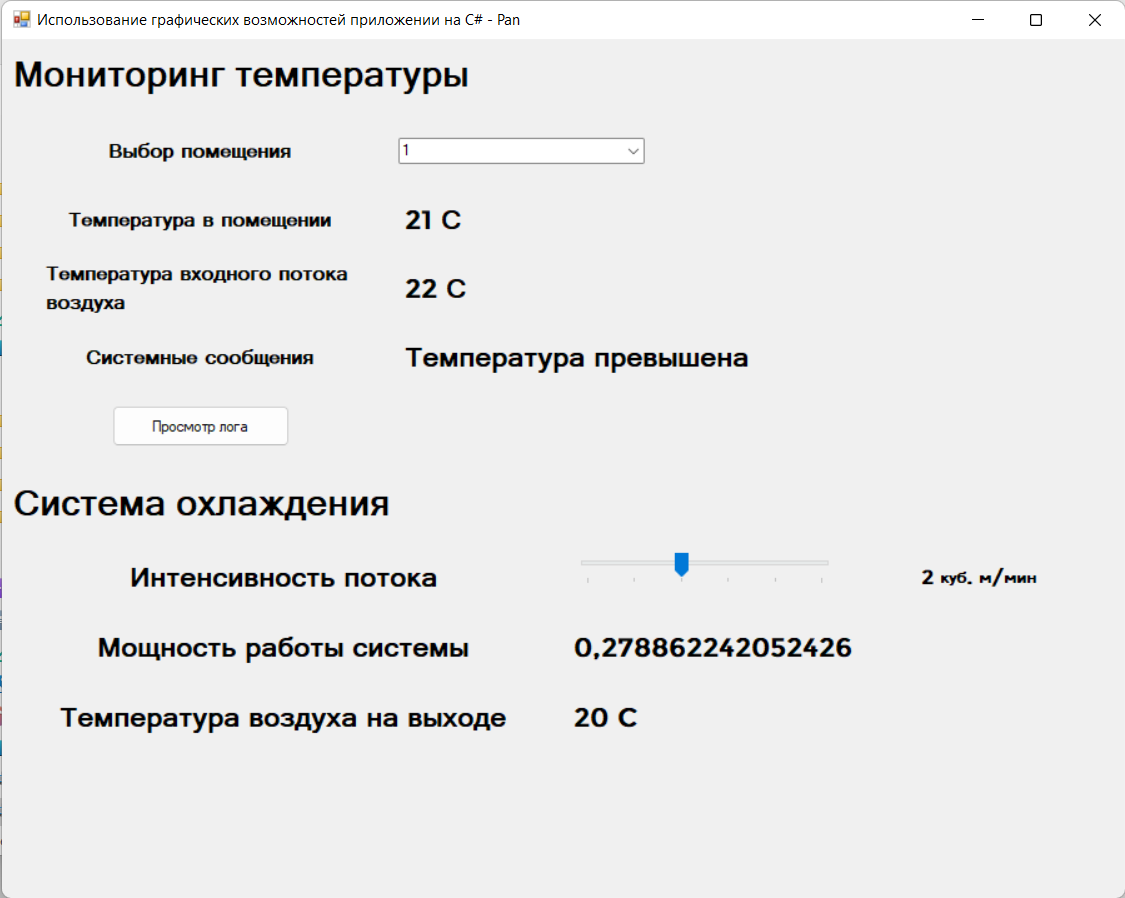
4. (Дополнительно) Реализовать отрисовку текущего состояния датчиков на схеме предприятия.

Ход работы

В ходе работы было разработа приложение на языке C# (Windows Forms) по работе с файлами и базой данных для хранения и обработки данных измерений. Реализовано хранение собранных данных в таблице БД и вывод (экспорт) данных за указанный период в текстовый файл.

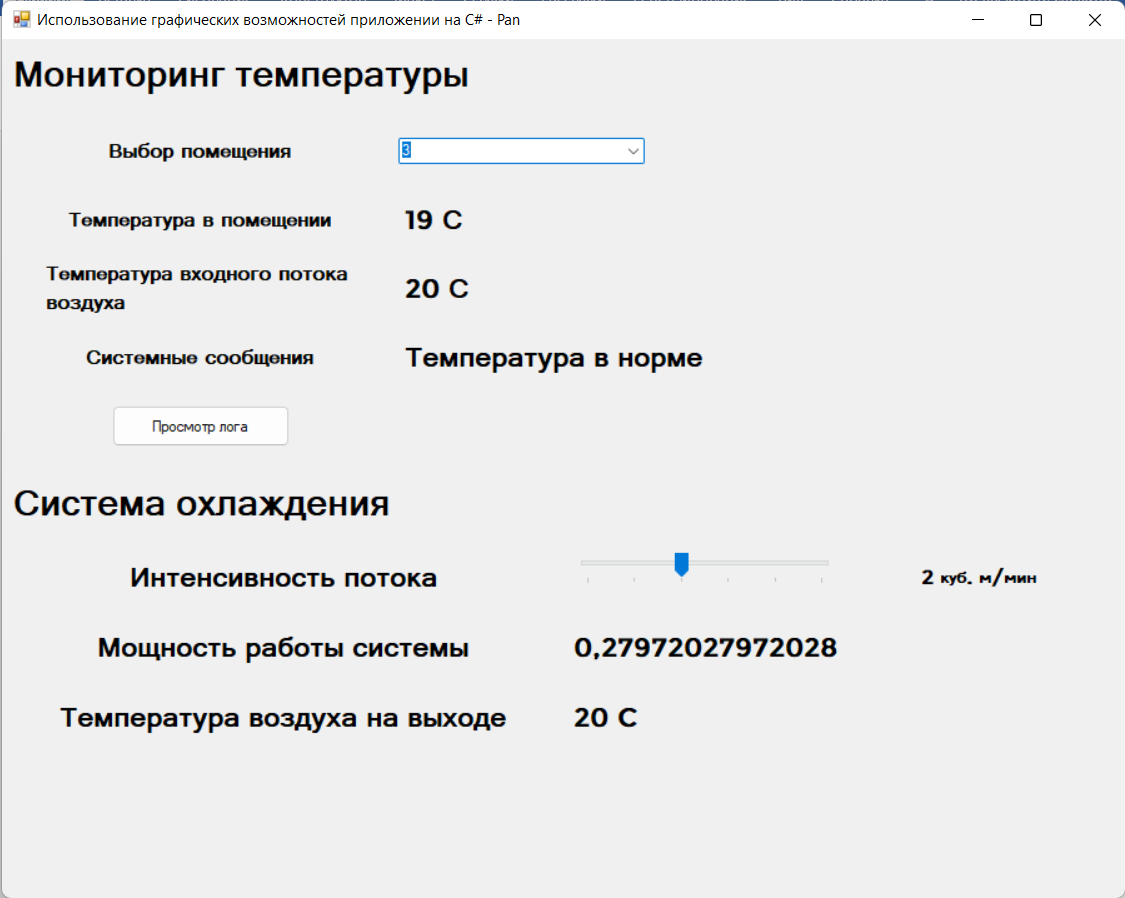
**Описание алгоритма работы программы**

При запуске программы открывается главное окно (Рисунок 1). На этом окне можно выбрать номер помещения, данные о температуре в котором будут отображены ниже.



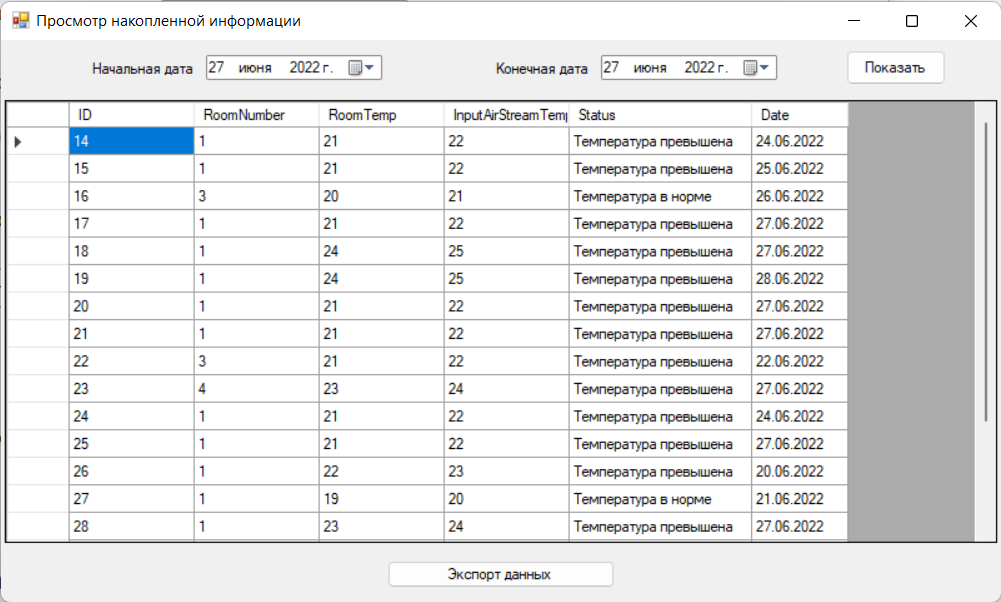
Рисунок

В нижней части окна находятся данные о системе охлаждения – интенсивность, мощность и температура воздуха на выходе. (Рисунок 2)



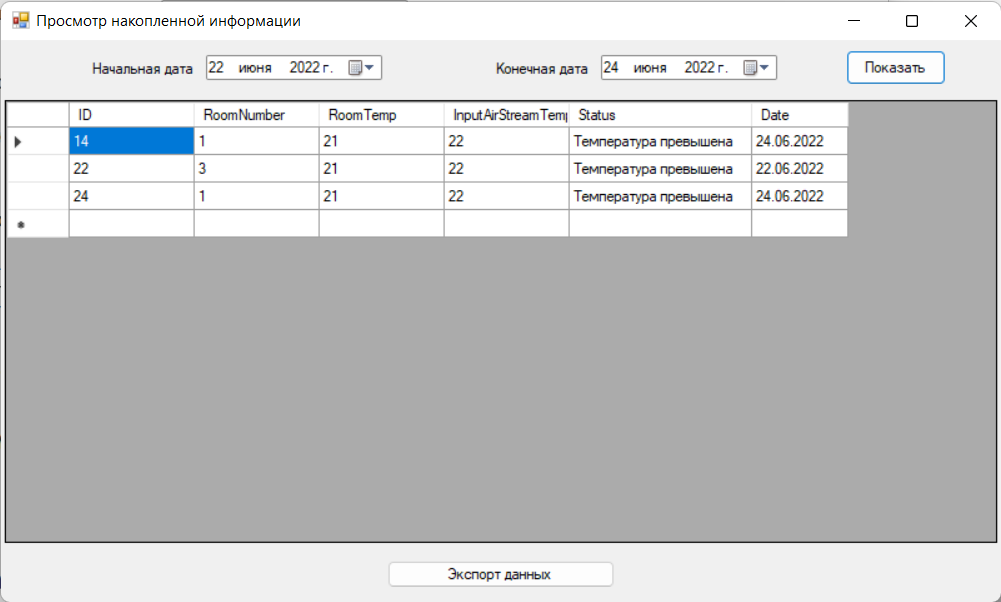
Рисунок

На Рисунке 3 изображено окно просмотра лога.



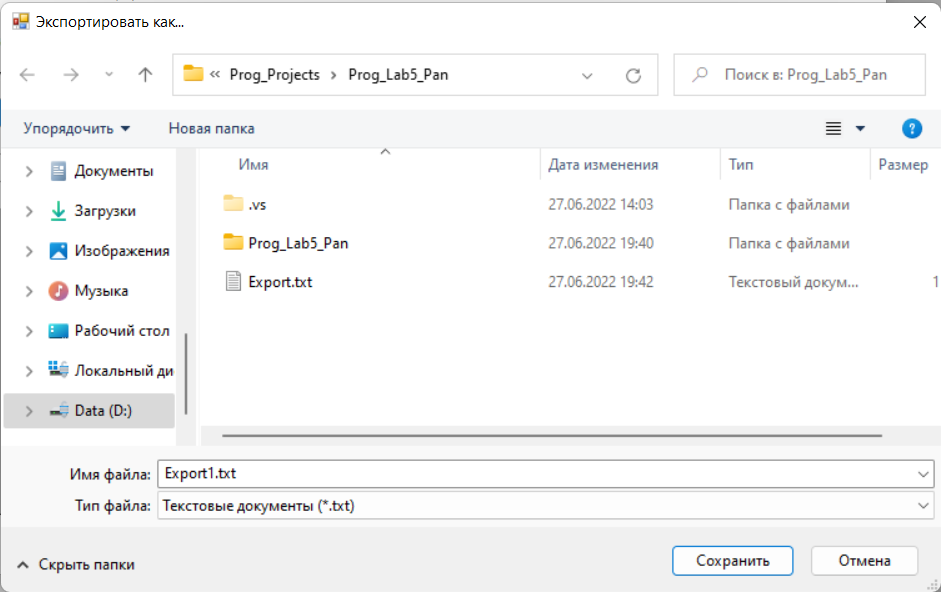
Рисунок

В этом окне можно отобразить данные за конкретный временной диапазон (Рис. 4)



Рисунок

По нажатию на кнопку «Экспорт» откроется окно для сохранения данных. (Рис. 5)



Рисунок

Проверка работы системы

При выборе номера помещения, на экран выводится температура воздуха в нем от 19 до 25 градусов. Нормой является 20 градусов. То, насколько градусов превышена температура относительно нормы, влияет на коэффициент при расчете мощности работы системы. Формула расчета мощности P = I/k, где I – интенсивность (в куб. м / мин), k – коэффициент.

Приложение А. Листинг кода

class DBActions

{

SqlConnection con = new SqlConnection(@"Data Source=(localdb)\ph\_study; Initial Catalog=ProgLabs; Integrated Security=true;");

Encoding utf8 = Encoding.UTF8;

public void openConnection()

{

if (con.State == System.Data.ConnectionState.Closed) con.Open();

}

public void closeConnection()

{

if (con.State == System.Data.ConnectionState.Open) con.Close();

}

public DataSet getAllData()

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = "SELECT \* FROM lab4\_ItemsTable";

command.Connection = con;

SqlDataAdapter da = new SqlDataAdapter(command);

DataSet data = new DataSet();

DataTable dt1 = data.Tables.Add();

DataTable dt2 = data.Tables.Add();

da.Fill(data.Tables[0]);

command.CommandText = "SELECT \* FROM lab4\_ProvidersTable";

da = new SqlDataAdapter(command);

da.Fill(data.Tables[1]);

closeConnection();

return data;

}

public DataTable getPoviderID(string ColName, string TableName)

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"SELECT {ColName} FROM {TableName}";

command.Connection = con;

SqlDataAdapter da = new SqlDataAdapter(command);

DataTable data = new DataTable();

closeConnection();

da.Fill(data);

return data;

}

public string getDataByID(string ID, string TableName, string ColName, string IDColName)

{

try

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"SELECT {ColName} FROM {TableName} WHERE {IDColName} = '{ID}'";

command.Connection = con;

SqlDataAdapter da = new SqlDataAdapter(command);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

string res = reader.GetValue(0).ToString();

reader.Close();

closeConnection();

return res;

}

catch (Exception e) { return $"error: {e}"; }

}

public DateTime getDatetimeByID(int ID, string TableName, string ColName)

{

try

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"SELECT {ColName} FROM {TableName} WHERE ID\_Item = '{ID}'";

command.Connection = con;

SqlDataAdapter da = new SqlDataAdapter(command);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

DateTime res = (DateTime)reader.GetValue(0);

reader.Close();

closeConnection();

return res;

}

catch { return DateTime.Now; }

}

public string editDataInTable1(int ID, string TableName, string IN, string IC, int PI, DateTime SD, int SA, int SC, int SN)

{

try

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"UPDATE {TableName} SET " +

$"ItemName = '{IN}', " +

$"ItemCode = '{IC}', " +

$"ProviderID = {PI}, " +

$"SupplyDate = '{SD}', " +

$"SupplyAmount = {SA}, " +

$"SupplyCost = {SC}, " +

$"StorageNumber = {SN} " +

$"WHERE ID\_Item = '{ID}'";

command.Connection = con;

command.ExecuteNonQuery();

closeConnection();

return "Данные успешно обновлены!";

}

catch { return "Произошла ошибка при обновлении данных"; }

}

public string editDataInTable2(int ID, string TableName, string PN, long PH)

{

try

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"UPDATE {TableName} SET " +

$"ProviderName = '{PN}', " +

$"PhoneNumber = {PH} " +

$"WHERE ID\_Provider = '{ID}'";

command.Connection = con;

command.ExecuteNonQuery();

closeConnection();

return "Данные успешно обновлены!";

}

catch { return "Произошла ошибка при обновлении данных"; }

}

// Q2

public DataTable getDataSetByID(string What, string TableName, string IDCol, string ID)

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"SELECT {What} FROM {TableName} WHERE {IDCol} = '{ID}'";

command.Connection = con;

SqlDataAdapter da = new SqlDataAdapter(command);

DataTable data = new DataTable();

da.Fill(data);

closeConnection();

return data;

}

// Q4

public string editItemCode(string TableName, string ID, string IC)

{

try

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"UPDATE {TableName} SET ItemCode = '{IC}' WHERE ItemName = '{ID}'";

command.Connection = con;

command.ExecuteNonQuery();

closeConnection();

return "Данные успешно обновлены!";

}

catch { return "Произошла ошибка при обновлении данных"; }

}

// Q1

public DataTable getDataSetFromDateTime(DateTime Sd, DateTime Ed, string SN)

{

openConnection();

SqlCommand command = new SqlCommand();

command.CommandText = $"SELECT \* FROM lab4\_ItemsTable WHERE (SupplyDate BETWEEN @Sd AND @Ed) AND (StorageNumber = '{SN}')";

command.Connection = con;

SqlParameter Sparameter = new SqlParameter("@Sd", Sd);

command.Parameters.Add(Sparameter);

SqlParameter Eparameter = new SqlParameter("@Ed", Ed);

command.Parameters.Add(Eparameter);

SqlDataAdapter da = new SqlDataAdapter(command);

DataTable data = new DataTable();

da.Fill(data);

closeConnection();

return data;

}