МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

МОРСКОЙ КОЛЛЕДЖ

Отчет по преддипломной практике .

Выполнил:

Студент гр. Пк-16/1

Пышногуб В.С.

Преподаватель:

Винокур Т.С

Севастополь 2019

Оглавление

[1.Получение задания и изучения предметной области 3](#_Toc27597944)

[1.1 Предметная область 3](#_Toc27597945)

[1.2 Постановка задачи 3](#_Toc27597946)

[2. Создание БД , настройка полей. 3](#_Toc27597947)

[3. Реализация взаимодействия программы , экспорта/импорта данных 4](#_Toc27597948)

[3.1 Импорт\экспорт 5](#_Toc27597949)

[4.Создание форм , настройка их отображения 6](#_Toc27597950)

[5.Реализация отбора . Реализация фильтрации записей 7](#_Toc27597951)

[5.1 Реализация отбора . 7](#_Toc27597952)

[5.2 Реализация фильтров 8](#_Toc27597953)

[6.Отладка , обработка ошибок . 8](#_Toc27597954)

[7.Создание обработчика ошибок экспорта\импорта . 9](#_Toc27597955)

[8.Список иллюстраций 10](#_Toc27597956)

# 1.Получение задания и изучения предметной области

В первый день практики мы прошли ТБ , расписались за нее и приступили к изучению предметной области будущего ПО. Предметной областью стало изучение биологических показателей морских обитателей , в частности :шпрот, хамсы, ставриды и др . Программа должна реализовывать работа с БД acess , производить автоматические вычисления , упрощать работу с большим количеством данных (импорт/экспорт данных). Сама программа необходима для работы с биологическими показателями , такими как :жирность , возраст , генерация , различные размеры и так далее . Такие данные необходимы для ведения статистики состояния популяции рыб а так же состояния полученного улова .

## Предметная область

Предметная область данного программного продукта тесно связанна с биологией южных морей. В частности , данный программный продукт должен упростить статистическую обработку информации по биологическим показателям морских обитателей южных морей. Для получения данных показателей , производится выборка экземпляров рыбы для проведения опытов, которые позволят получить ее показатели . Пример опытов . Например , есть у нас выборка из 100 рыб , которая делится на группы по 10 рыб . Каждая рыба измеряется , для каждой группы вычисляются максимальный , минимальный размер рыбы , а так же среднее значение . Для каждой группы устанавливается вес максимальны , минимальный , средний . После , данная рыба , иногда , отправляется на получение значения жирности через опыты . Рыбу перемалывают в фарш , взвешивают , потом сушат , снова взвешивают , потом специальными реактивами растворяют жиры . В итоге , получается среднее значение жирности . По результатам , из данной выборки получают средние статистические данные , относящиеся к данному улову . При проведении анализа каждой особи рыбы , происходит установка ее возраста , года генерации , пола и так далее . Такие данные позволяют отслеживать состояние рыбы , прибыльность улова и вообще общие показания статистики по всему черному морю .

## Постановка задачи

Требовалось создать программный продукт , который упростит и автоматизирует работу с большим количеством информации , позволит вести статистику и реализует удобство работы . Краткие требования : это должна быть СУБД с реализацией импорта\экспорта данных , реализация табличного ввода информации , ее хранения , изменения , реализация настройки отображения полей .

# 2. Создание БД , настройка полей.

Для начала работы по созданию программы , необходима была опорная БД . Для ее создания был получен файл EXEL с тестовыми данными , которые позволят настроить поля таблиц и настроить взаимодействие ключей . Была создана БД с тремя таблицами . Каждая таблица была настроена , было настроено их взаимодействие между собой .

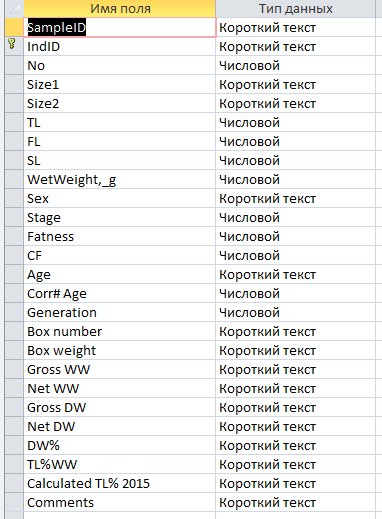


Рисунок 1 - конструктор таблицы "БА"

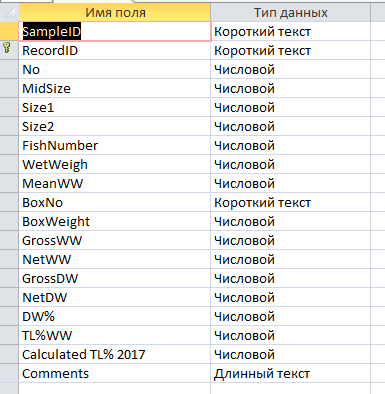


Рисунок 2 - конструктор таблицы "пробы"

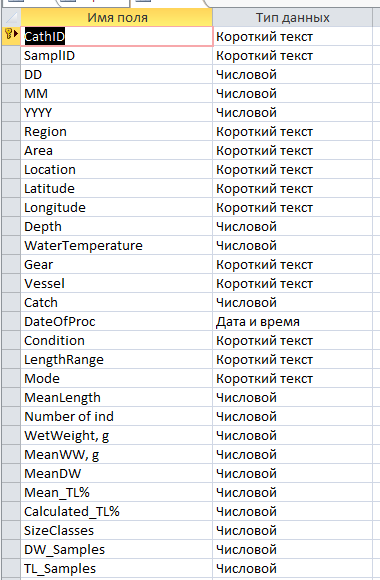
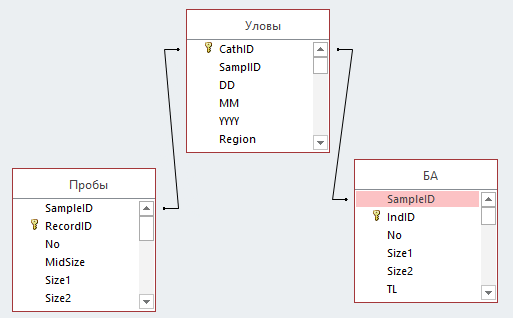
 

Рисунок 3 - конструктор таблицы "уловы" Рисунок 4 - схема данных БД

Все поля таблиц были настроены согласно полученному файлу EXEL .

# 3. Реализация взаимодействия программы , экспорта/импорта данных

Для реализации взаимодействия ПО с БД использовался «Источник данных» в Visual Studio . «Источник данных» позволяет реализовать простое взаимодействие программы с файлом БД посредством локального SQL сервера. Так же данная реализация позволяет использовать «DataSet» и «TableAdapter» , что в разы упрощает работу с БД . Все данные буду получаться через эти два объекта. Для проверки работоспособности были созданы базовые формы и подключена БД к программе. При добавлении объектов , Visual Studio сам создал запрос на загрузку данных в таблицу , что позволило в дальнейшем начать разработку импорта\экспорта .

## 3.1 Импорт\экспорт

Импорт\экспорт был реализован , путем построчного считывания текстового файла (csv файла ) . Если кратко , то при загрузке данных , программа считывает строку , строку делит на значения по правилу и записывает данные значения в БД , после чего приступает к считыванию следующей строки . И так до конца файла . Если произойдет ошибка – программа не будет грузить информацию в БД , но , создаст файл со списком ошибок .

Выгрузка была реализована циклически , по каждой строке в таблице .Кратко :программа считывает всю строку , записывает ее с разделителями , переходит на следующую строку , и так , пока есть строки .

Ниже приведены фрагменты кода импорта и экспорта

Рисунок 5- фрагмент кода импорта

public string Error = "";

public bool bl = false;

ArrayList erlli = new ArrayList();

ArrayList erll = new ArrayList();

private void импортToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpFile.Filter = "csv files (\*.csv)|\*.csv|All files (\*.\*)|\*.\*";

if (OpFile.ShowDialog() == DialogResult.OK)

{

erlli.Clear();

erll.Clear();

ArrayList lines = new ArrayList();

string ll; string str = ""; int count = 0;

string patch = OpFile.FileName;

StreamReader sr = new StreamReader(patch, Encoding.Default);

int li = 0;

while ((ll = sr.ReadLine()) != null)

{

li++;

for (int i = 0; i < ll.Length; i++)

{

if ((ll[i] != ';') || (ll[i] == (Convert.ToChar(13))))

{

str = str + ll[i];

}

else

{

if (str == "")

{

lines.Add(null);

}

else

lines.Add(str);

str = "";

count++;

}

}

// пробыDataGridView.Rows.Add(lines[0], lines[1], lines[2], lines[3], lines[4], lines[5], lines[6], lines[7], lines[8], lines[9], lines[10], lines[11], lines[12], lines[13], lines[14], lines[15], lines[16], lines[17]);

//пробыTableAdapter.Adapter.InsertCommand.CommandText = "INSERT INTO Пробы VALUES("+Convert.ToString(lines[0])+")";

try

{

if (lines[2] == null) lines[2] = -999;

if (lines[3] == null) lines[3] = -999;

if (lines[4] == null) lines[4] = -999;

if (lines[10] == null) lines[10] = -999;

if (lines[11] == null) lines[11] = -999;

if (lines[14] == null) lines[14] = -999;

if (lines[19] == null) lines[19] = -999;

if (lines[20] == null) lines[20] = -999;

if (lines[21] == null) lines[21] = -999;

if (lines[22] == null) lines[22] = -999;

if (lines[23] == null) lines[23] = -999;

if (lines[24] == null) lines[24] = -999;

if (lines[25] == null) lines[25] = -999;

if (lines[26] == null) lines[26] = -999;

if (lines[27] == null) lines[27] = -999;

if (lines[28] == null) lines[28] = -999;

if (lines[31] == null) lines[31] = -999;

}catch

{ }

try

{

уловыTableAdapter.Insert(Convert.ToString(lines[0]),

Convert.ToString(lines[1]),

Convert.ToInt16(lines[2]),

Convert.ToInt16(lines[3]),

Convert.ToInt16(lines[4]),

Convert.ToString(lines[5]),

Convert.ToString(lines[6]),

Convert.ToString(lines[7]),

Convert.ToString(lines[8]),

Convert.ToString(lines[9]),

Convert.ToDouble(lines[10]),

Convert.ToDouble(lines[11]),

Convert.ToString(lines[12]),

Convert.ToString(lines[13]),

Convert.ToDouble(lines[14]),

Convert.ToDateTime(lines[15]),

Convert.ToString(lines[16]),

Convert.ToString(lines[17]),

Convert.ToString(lines[18]),

Convert.ToDouble(lines[19]),

Convert.ToDouble(lines[20]),

Convert.ToDouble(lines[21]),

Convert.ToDouble(lines[22]),

Convert.ToDouble(lines[23]),

Convert.ToDouble(lines[24]),

Convert.ToDouble(lines[25]),

Convert.ToDouble(lines[26]),

Convert.ToDouble(lines[27]),

Convert.ToDouble(lines[28]),

Convert.ToString(lines[29]),

Convert.ToString(lines[30]),

Convert.ToDouble(lines[31]),

Convert.ToString(lines[32]),

Convert.ToString(lines[33]),

Convert.ToString(lines[34])

);

}

catch (System.Exception ex)

{

//System.Windows.Forms.MessageBox.Show(ex.Message, "Ошибка");

//break;

erll.Add("Ошибка записи в поле " + Convert.ToString(li) + ", RecordlID = \"" + lines[1] + "\"" + " исходная строка : \n");

erlli.Add("{" + ll + "}");

//Error = Error + "Ошибка записи в поле " + Convert.ToString(li) + ", ID Пробы = \"" + lines[1] + "\"" + "\n";

bl = true;

}

lines.Clear();

str = "";

}

sr.Close();

}

if (bl == true)

{

string sl = DateTime.Now.ToString("dd.MM.yy HH\_mm\_ss");

string patch = "Ошибки импорта/Ошибки импорта " + sl + ".csv";

System.Windows.Forms.MessageBox.Show("При импорте произошла ошибка. Для более подробной информации произведите анализ файла \"" + patch + "\" в анализаторе ошибок ", "Список строк с ошибками");

using (StreamWriter sw = new StreamWriter(patch, false, System.Text.Encoding.Default))

{

sw.WriteLine("Ошибка импорта в таблицу ^Уловы^ . Ошибочные строки :");

for (int i = 0; i < erll.Count; i++)

{

sw.WriteLine(erll[i]);

sw.WriteLine(erlli[i]);

}

sw.Close();

}

// System.Windows.Forms.MessageBox.Show(Error, "Список строк с ошибками");

bl = false;

}

else

{

if (bl != true) this.tableAdapterManager.UpdateAll(this.тестовая\_бд\_DataSet);

try

{

//this.пробыTableAdapter.Svaz(this.тестовая\_бд\_DataSet.Пробы, Convert.ToString(mainform.уловыDataGridView[0, mainform.уловыDataGridView.CurrentCell.RowIndex].Value));

this.уловыTableAdapter.FillBy(this.тестовая\_бд\_DataSet.Уловы, Convert.ToString(mainform.indt[mainform.listBox1.SelectedIndex]));

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message, "Ошибка");

}

}

}

Для каждой таблицы , код импорта\экспорта отличается , но , принцип один и тот же .

Рисунок 6 - фрагмент кода экспорта

private void экспортToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFile.Filter = "csv files (\*.csv)|\*.csv|All files (\*.\*)|\*.\*";

if (SaveFile.ShowDialog() == DialogResult.OK)

{

string path = SaveFile.FileName;

if (path != null)

{

try

{

using (StreamWriter sw = new StreamWriter(path, false, System.Text.Encoding.Default))

{

for (int i = 0; i < уловыDataGridView.Rows.Count - 1; i++)

{

for (int j = 0; j <= 34; j++)

{

{

if ((Convert.ToString(уловыDataGridView[j, i].Value) != "") && (Convert.ToString(уловыDataGridView[j, i].Value) != "-999"))

sw.Write(Convert.ToString(уловыDataGridView[j, i].Value));

else sw.Write(Convert.ToString(null));

sw.Write(";");

if (j == 34)

{

sw.Write(" ;");

}

}

}

sw.WriteLine("");

}

sw.Close();

}

}

catch (Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

}

}

# 4.Создание форм , настройка их отображения

Для удобства работы с программным продуктом требовалось создать удобные формы. Формы в данном ПО бывают двух типов :табличные и форма каждого элемента . Были сделаны только табличные формы , а так же , создана настройка отображения табличных частей на форме . Все формы обзавелись верхним меню и формой настройки отображения . Пример форм приведён ниже

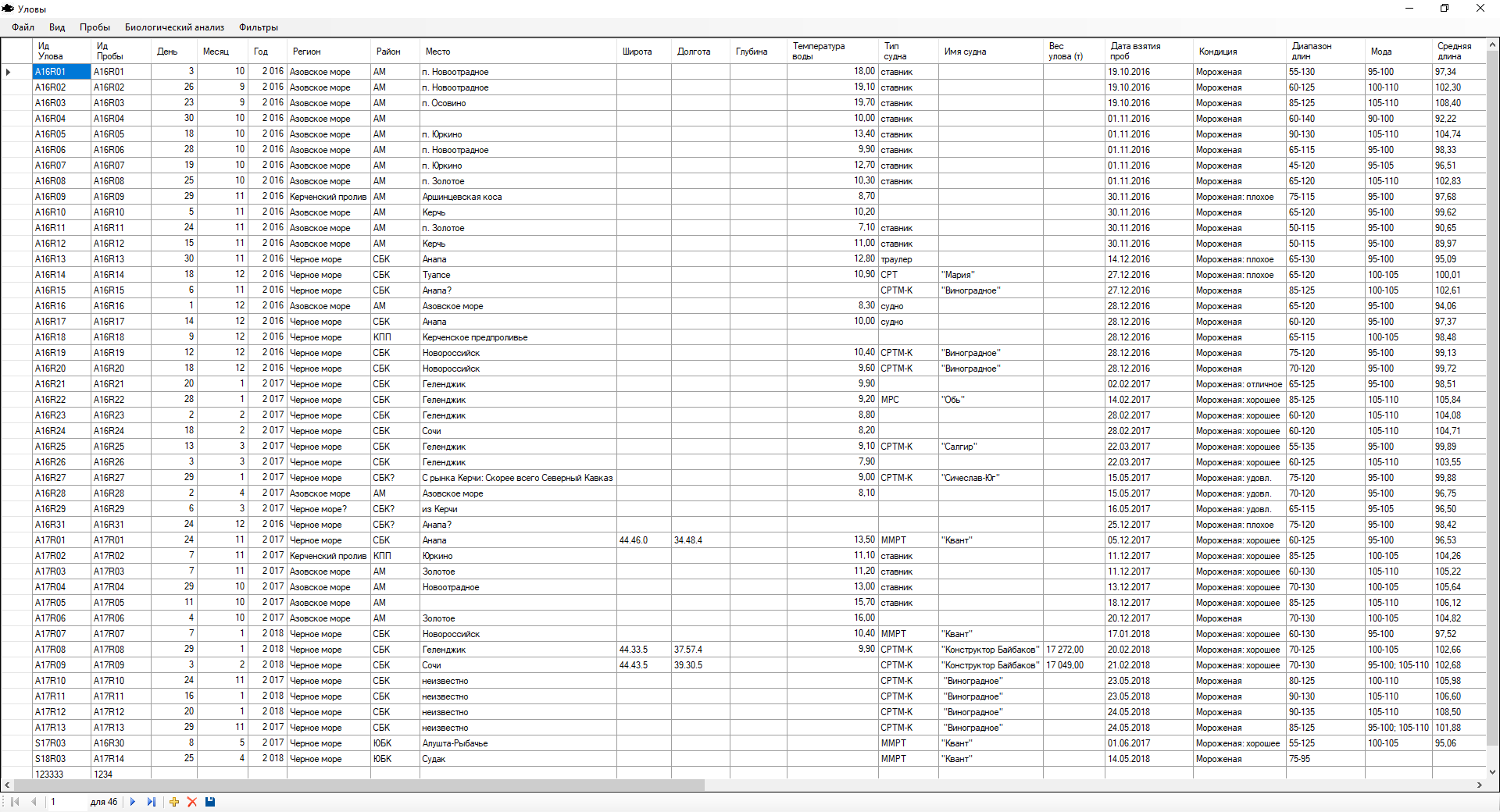
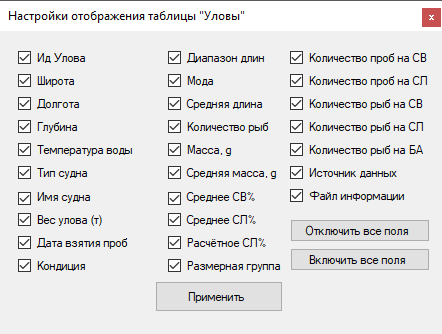


Рисунок 7 - форма "уловы"



# 5.Реализация отбора . Реализация фильтрации записей

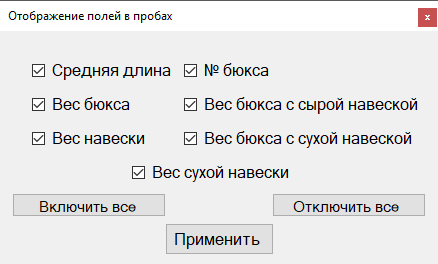
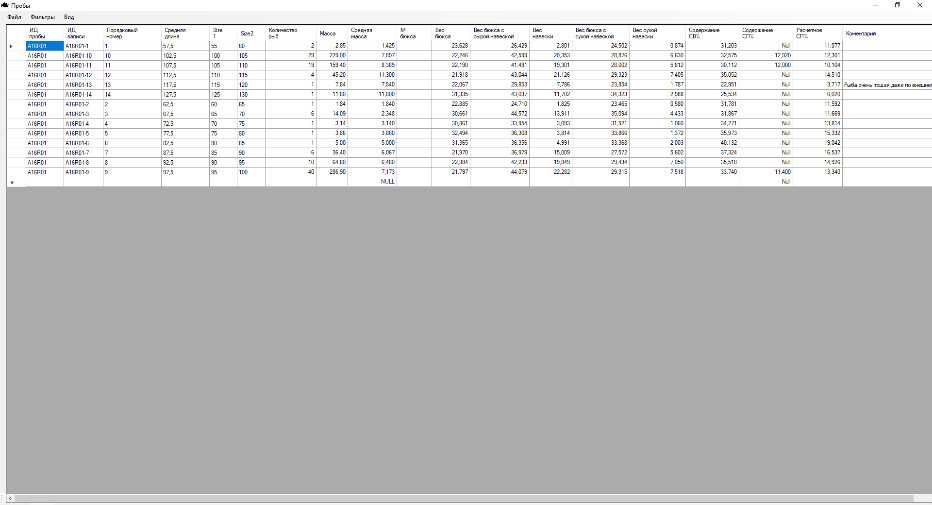


Рисунок 8- форма "пробы"

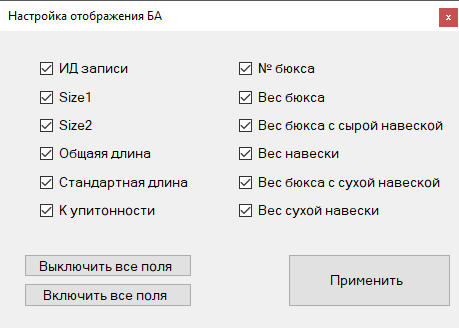
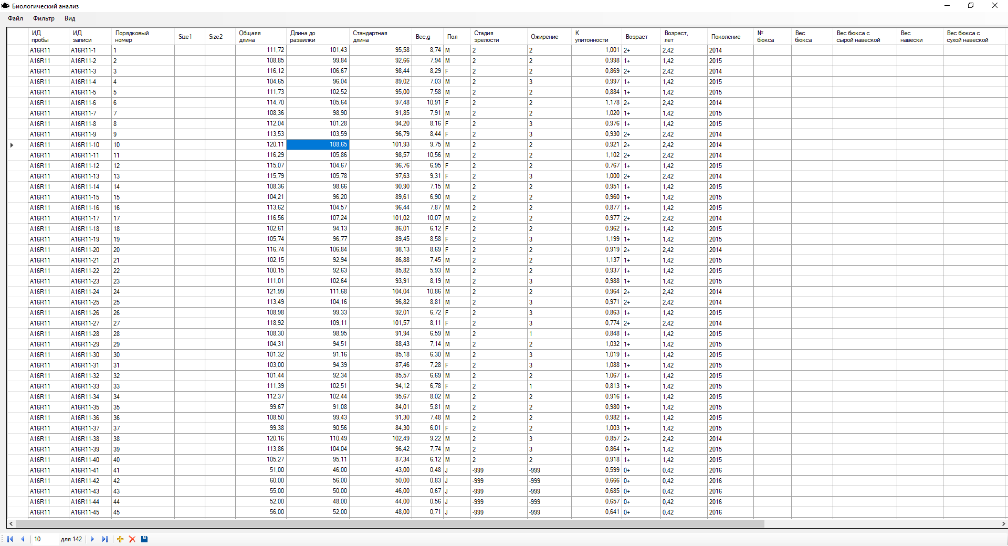


Рисунок 9 - форма "БА"

Для полноценной и удобной работы с СУБД требуется реализовать возможность фильтрации записей , а так же просмотра связанных с данной записей информации в других таблицах ( реализовать отбор данных по выбранной записи ).

## 5.1 Реализация отбора .

Отбор был реализован путем создания SQL команд на выборку информации по заданным параметрам . Ограничение запроса – открытие сразу 80 разных записей ( можно ввести до 80 разных ключевых поле , по которым происходит поиск записей ) . Так же , поскольку реализован отбор до 80 записей , был создан обработчик , который позволяет при зажатой кнопке «CTRL» выделять записи и по ним открывать информацию . Пример кода , использующегося для реализации данной функции приведён ниже .

Рисунок 10 - реализация запоминания ключевого поля при зажатой "CTRL"

public string key;

private void уловыDataGridView\_KeyDown(object sender, KeyEventArgs e)

{

key = e.KeyCode.ToString();

// cells.Clear();

// MessageBox.Show(e.KeyCode.ToString());

}

public int k = 1;

private void уловыDataGridView\_Click(object sender, EventArgs e)

{

if (key == "ControlKey")

{

try { cells[k] = (Convert.ToString(уловыDataGridView[1, уловыDataGridView.CurrentCell.RowIndex].Value)); }

catch { }

k++;

}

else

{

try

{

cells[0] = (Convert.ToString(уловыDataGridView[1, уловыDataGridView.CurrentCell.RowIndex].Value));

}

catch { }

for (int i = 1; i <= 80; i++)

cells[i] = (999999);

k = 1;

}

}

private void уловыDataGridView\_KeyUp(object sender, KeyEventArgs e)

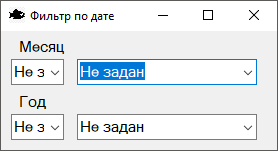
{

key = "";

}

## 5.2 Реализация фильтров

Вся фильтрация была построена на SQL командах . Для запуска фильтров были созданы мини формы , который позволяют выбрать тип фильтра и задать ему значение . Сложным здесь было то , что для каждого фильтра надо было написать SQL запрос , который учитывал бы не только фильтр но и то , какие записи сейчас открыты .



SELECT CathID, SamplID, DD, MM, YYYY, Region, Area, Location, Latitude, Longitude, Depth, WaterTemperature, Gear, Vessel, Catch, DateOfProc, Condition, LengthRange, Mode, MeanLength, [Number of ind], [WetWeight, g],

[MeanWW, g], MeanDW, [Mean\_TL%], [Calculated\_TL%], SizeClasses, DW\_Samples, TL\_Samples, DW\_Ind, TL\_Ind, BioAnalyses\_Ind, Comments, DataSource, DataFile

FROM Уловы

WHERE (DD = ?) OR (MM = ?) OR (YYYY = ?)

Рисунок 11 - пример формы фильтра и его SQL запроса

# 6.Отладка , обработка ошибок .

В процессе создания программного продукта , а так же при его работе возможно большое количество ошибок . Для того , что бы эти ошибки не вызывали сбоев работы программного обеспечения , а так же , что бы работал полный функционал , была выполнена отладка . Так же , в тех местах , где можно это сделать , были написаны обработчики ошибок , которые предупреждают пользователей о возникновении ошибки , но не завершают работу программы .

# 7.Создание обработчика ошибок экспорта\импорта .

При экспорте\импорте возможно возникновение ошибок . При появлении таких , программа создает файл с списком строк , ф котором возникли ошибки , так же , в файле хранится название таблицы . Данный фал нужен для того , что бы потом через обработчик ошибок , узнать , в каком поле , какая ошибка .

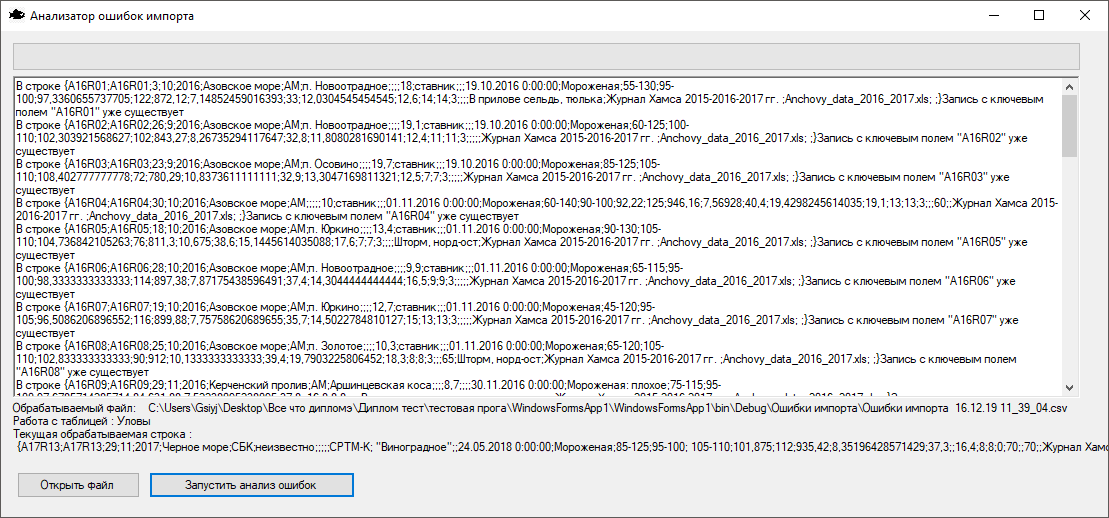


Рисунок 12 - форма обработчика ошибок

Интерфейс прост до безобразия . Вся обработка происходит автоматически , после согласия с предупреждением . По итогу работы , создается файл с отчетом , в котором описаны ошибки по каждой строке .

public string ErDT(string tb, ArrayList lines, int countlin)

{

int idnt = 3; int colCount = 0; ; string er = "Ошибок нет"; bool erdet = false;

if (tb == "Уловы") { idnt = 0; colCount = 35; }

if (tb == "Пробы") { idnt = 1; colCount = 19; }

if (tb == "БА") { idnt = 2; colCount = 26; }

if (countlin != colCount)

{

er = "Колличество ячеек в строке не совпадает с количесвом ячеек таблицы , в которую происходит импорт .";

if (countlin == 35)

{

er += "Возможно данные данные из таблицы \"Уловы\"";

}

if (countlin == 19)

{

er += "Возможно данные данные из таблицы \"Пробы\"";

}

if (countlin == 26)

{

er += "Возможно данные данные из таблицы \"БА\"";

}

}

else

{

if (idnt == 0)

{

er = "";

for (int j = 0;j< уловыDataGridView.Rows.Count; j++)

{

if (Convert.ToString(lines[0])== Convert.ToString(уловыDataGridView[0,j].Value))

{

er += "Запись с ключевым полем \"" + Convert.ToString(lines[0]) + "\" уже существует ";

}

}

try

{

short kl = Convert.ToInt16(lines[2]);

}

catch (Exception ex)

{

er = "Ошибка в поле 2 :" + ex.Message + ". ";

}

try

{

short kl = Convert.ToInt16(lines[3]);

}

catch (Exception ex)

{

er = "Ошибка в поле 3 :" + ex.Message + ". ";

}

try

{

short kl = Convert.ToInt16(lines[4]);

}

catch (Exception ex)

{

er = "Ошибка в поле 4 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[10]);

}

catch (Exception ex)

{

er = "Ошибка в поле 10 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[11]);

}

catch (Exception ex)

{

er = "Ошибка в поле 11 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[14]);

}

catch (Exception ex)

{

er = "Ошибка в поле 14 :" + ex.Message + ". ";

}

try

{

DateTime kl = Convert.ToDateTime(lines[15]);

}

catch (Exception ex)

{

er = "Ошибка в поле 15 :" + ex.Message + ". ";

}

for (int j= 19; j<= 28; j++)

{

try

{

double kl = Convert.ToDouble(lines[j]);

}

catch (Exception ex)

{

er = "Ошибка в поле "+Convert.ToString(j)+" :" + ex.Message + ". ";

}

}

try

{

double kl = Convert.ToDouble(lines[31]);

}

catch (Exception ex)

{

er = "Ошибка в поле 31 :" + ex.Message + ". ";

}

}

if (idnt == 1)

{

er = "";

for (int j = 0; j < пробыDataGridView.Rows.Count; j++)

{

if (Convert.ToString(lines[1]) == Convert.ToString(пробыDataGridView[1, j].Value))

{

er += "Запись с ключевым полем \"" + Convert.ToString(lines[1]) + "\" уже существует ";

}

}

try

{

double kl = Convert.ToDouble(lines[2]);

}

catch (Exception ex)

{

er = "Ошибка в поле 2 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[3]);

}

catch (Exception ex)

{

er = "Ошибка в поле 3 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[7]);

}

catch (Exception ex)

{

er = "Ошибка в поле 7 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[8]);

}

catch (Exception ex)

{

er = "Ошибка в поле " + Convert.ToString(8) + " :" + ex.Message + ". ";

}

for (int j = 10; j <= 17; j++)

{

try

{

double kl = Convert.ToDouble(lines[j]);

}

catch (Exception ex)

{

er = "Ошибка в поле " + Convert.ToString(j) + " :" + ex.Message + ". ";

}

}

}

if (idnt == 2)

{

er = "";

for (int j = 0; j < бАDataGridView.Rows.Count; j++)

{

if (Convert.ToString(lines[1]) == Convert.ToString(бАDataGridView[1, j].Value))

{

er += "Запись с ключевым полем \"" + Convert.ToString(lines[1]) + "\" уже существует ";

}

}

try

{

short kl = Convert.ToInt16(lines[2]);

}

catch (Exception ex)

{

er = "Ошибка в поле 2 :" + ex.Message + ". ";

}

for (int j = 5; j <= 8; j++)

{

try

{

double kl = Convert.ToDouble(lines[j]);

}

catch (Exception ex)

{

er = "Ошибка в поле " + Convert.ToString(j) + " :" + ex.Message + ". ";

}

}

try

{

short kl = Convert.ToInt16(lines[10]);

}

catch (Exception ex)

{

er = "Ошибка в поле 10 :" + ex.Message + ". ";

}

try

{

short kl = Convert.ToInt16(lines[11]);

}

catch (Exception ex)

{

er = "Ошибка в поле 11 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[12]);

}

catch (Exception ex)

{

er = "Ошибка в поле 12 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[14]);

}

catch (Exception ex)

{

er = "Ошибка в поле 14 :" + ex.Message + ". ";

}

try

{

double kl = Convert.ToDouble(lines[15]);

}

catch (Exception ex)

{

er = "Ошибка в поле 15 :" + ex.Message + ". ";

}

}

}

return(er);

}

ArrayList cells = new ArrayList();

Рисунок 13 - часть кода обработчика ошибок

# 8.Список иллюстраций

[Рисунок 1 - конструктор таблицы "БА" 4](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597931)

[Рисунок 2 - конструктор таблицы "пробы" 4](#_Toc27597932)

[Рисунок 3 - конструктор таблицы "уловы" Рисунок 4 - схема данных БД 4](#_Toc27597933)

[Рисунок 5- фрагмент кода импорта 5](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597934)

[Рисунок 6 - фрагмент кода экспорта 6](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597935)

[Рисунок 7 - форма "уловы" 6](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597936)

[Рисунок 8- форма "пробы" 7](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597937)

[Рисунок 9 - форма "БА" 7](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597938)

[Рисунок 10 - реализация запоминания ключевого поля при зажатой "CTRL" 8](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597939)

[Рисунок 11 - пример формы фильтра и его SQL запроса 8](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597940)

[Рисунок 12 - форма обработчика ошибок 9](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597941)

[Рисунок 13 - часть кода обработчика ошибок 9](file:///C:\Users\Gsiyj\Desktop\Все%20что%20дипломэ\Отчёт%20по%20практике%20.docx#_Toc27597942)