#include <iostream>

#include <string>

#include <windows.h>

#include <fstream>

#include "Fail.h"

int main ()

{

}

//1)----------------------------------------------------------------------------------------------

private void set\_mode(bool mod)

{

if (mod == true)

{

label1.Enabled = true;

button1.Enabled = true;

button2.Enabled = true;

button3.Enabled = false;

button4.Enabled = false;

}

else

{

label1.Enabled = false;

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = true;

button4.Enabled = true;

}

}

// Ðåøåíèå

private void set\_mode(bool mod)

{

if (mod == true)

{

label1.Enabled = mod;

button1.Enabled = mod;

button2.Enabled = mod;

button3.Enabled = !mod;

button4.Enabled = !mod;

}

else

{

label1.Enabled = !mod;

button1.Enabled = !mod;

button2.Enabled = !mod;

button3.Enabled = mod;

button4.Enabled = mod;

}

}

//2)----------------------------------------------------------------------------------------------

switch (driver.Status)

{

case ClientStatus.Unknown:

return m\_driverStatusNames[ClientStatus.Unknown];

case ClientStatus.Free:

return m\_driverStatusNames[ClientStatus.Free];

case ClientStatus.Busy:

return m\_driverStatusNames[ClientStatus.Busy];

case ClientStatus.InWay:

return m\_driverStatusNames[ClientStatus.InWay];

case ClientStatus.Work:

return m\_driverStatusNames[ClientStatus.Work];

case ClientStatus.Break:

return m\_driverStatusNames[ClientStatus.Break];

case ClientStatus.Alarm:

return m\_driverStatusNames[ClientStatus.Alarm];

}

// Ðåøåíèå

switch (driver.Status)

{

case ClientStatus.Unknown:

case ClientStatus.Free:

case ClientStatus.Busy:

case ClientStatus.InWay:

case ClientStatus.Work:

case ClientStatus.Break:

case ClientStatus.Alarm:

return m\_driverStatusNames[driver.Status];

}

//3)----------------------------------------------------------------------------------------------

uint i;

…

if (i.ToString().Length == 1)

{

...

}

// Ðåøåíèå

uint i;

…

if (i < 10)

{

...

}

//4)----------------------------------------------------------------------------------------------

string destination = null;

for (int i = 0; i < 13; i++)

destination += source[i];

// Ðåøåíèå

string destination = null;

destination = source.SubString(0, 13);

//5)----------------------------------------------------------------------------------------------

Bool IsNumber(string str) {

return (str.Replace("0", "").Replace("1", "").Replace("2", "").Replace

("3", "").Replace("4", "").Replace("5", "").Replace("6", "").Replace

("7", "").Replace("8", "").Replace("9", "").Length == 0);

}

// Ðåøåíèå

Bool IsNumber(string str) {

for (int i = 0; i < 10; i++)

{

str.Replace(Convert.ToString(i), "")

}

return (str.Length == 0);

}

//6)----------------------------------------------------------------------------------------------

foreach(DirectoryInfo dir in dirs.GetDirectories())

{

//create folder{16}

stream.Write(new byte[]{ (byte)NetworkMessage.MakeDir }, 0, 1);

stream.Read(new byte[1], 0, 1);

stream.Write(BitConverter.GetBytes(Encoding.UTF8.GetBytes(

SubFolder.Replace('\\', '/') + dir.Name.Replace('\\', '/')).Length), 0, 4);

stream.Write(Encoding.UTF8.GetBytes(SubFolder.Replace('\\', '/') +

dir.Name.Replace('\\', '/')), 0,

Encoding.UTF8.GetBytes(SubFolder.Replace('\\', '/') +

dir.Name.Replace('\\', '/')).Length);

//send folder name

stream.Read(new byte[1], 0, 1);//Ok

}

// Ðåøåíèå

Directory.CreateDirectory("Ïóòü ê ïàïêå");

//7)----------------------------------------------------------------------------------------------

String[] days = new String[7];

for (int i = 0; i < 7; i++) {

switch (i) {

default:

case 0:

days[i] = "Monday";

break;

case 1:

days[i] = "Tuesday";

break;

case 2:

days[i] = "Wednesday";

break;

case 3:

days[i] = "Thursday";

break;

case 4:

days[i] = "Friday";

break;

case 5:

days[i] = "Saturday";

break;

case 6:

days[i] = "Sunday";

break;

}

}

// Ðåøåíèå

String[] days = new String[7];

switch (i)

{

default:

case 0:

days[0] = "Monday";

break;

case 1:

days[1] = "Tuesday";

break;

case 2:

days[2] = "Wednesday";

break;

case 3:

days[3] = "Thursday";

break;

case 4:

days[4] = "Friday";

break;

case 5:

days[5] = "Saturday";

break;

case 6:

days[6] = "Sunday";

break;

}

}

//8)----------------------------------------------------------------------------------------------

DateTime dt = DateTime.Now;

string h = dt.Hour.ToString().PadLeft(2, '0');

string m = dt.Minute.ToString().PadLeft(2, '0');

string s = dt.Second.ToString().PadLeft(2, '0');

Console.WriteLine("--" + h + ":" + m + ":" + s + "--");

// Ðåøåíèå

DateTime dt = DateTime.Now;

Console.WriteLine("--" + dt.Hour.ToString().PadLeft(2, '0') + ":" + dt.Minute.ToString().PadLeft(2, '0') + ":" + dt.Second.ToString().PadLeft(2, '0') + "--");

Console.WriteLine("Hello World!");

//9)----------------------------------------------------------------------------------------------

return ((int)(Counter / 2) != Counter / 2.00 && Counter != 0);

// Ðåøåíèå

return (Counter % 2 != Counter / 2 && Counter != 0);

//10)----------------------------------------------------------------------------------------------

if (Connected == 0))

{

rez = setup();

fl\_end = true; // âûõîä

}

else

fl\_end = true;

// Ðåøåíèå

if (Connected == 0))

{

rez = setup();

}

fl\_end = true; // âûõîä

//11)----------------------------------------------------------------------------------------------

List<int> arr = new List<int>();

List<int> tmpArr = new List<int>();

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

{

if (arr[i] > 100)

{

}

else

tmpArr.Add(arr[i]);

}

arr = tmpArr;

// Ðåøåíèå

List<int> intList = IntListFactory.GetRandom();

intList.RemoveAll(delegate(int number) { return number > 100; });

//12)----------------------------------------------------------------------------------------------

var ids = form.Keys;

if (ids.Length == 0 || ids.Length > 1) { throw Exception; }

// Ðåøåíèå

var ids = form.Keys;

if (ids.Length != 1)

throw Exception;

//13)----------------------------------------------------------------------------------------------

//Use the Name field

string[] nameParts = customer.Name.Split(' ');

string firstName = nameParts[0];

string lastName = customer.Name.Replace(nameParts[0], "").TrimStart(' ');

// Ðåøåíèå

string[] nameParts = customer2.Name.Split(" ");

//14)----------------------------------------------------------------------------------------------

foreach(string id in sourceIDs.Split(new char[] { ',' },StringSplitOptions.RemoveEmptyEntries))

{

sourceId = Convert.ToInt32(id);

break;

}

// Ðåøåíèå

foreach(string id in sourceIDs.Split(",", StringSplitOptions.RemoveEmptyEntries))

{

sourceId = Convert.ToInt32(id);

break;

}

//15)----------------------------------------------------------------------------------------------

public string generateEMail()

{

string res;

int i = PersonName.IndexOf(" ");

char[] str1 = new char[i];

PersonName.CopyTo(0, str1, 0, i);

string str11 = new string(str1);

char[] str2 = new char[PersonName.Length - i - 1];

PersonName.CopyTo(i + 1, str2, 0, PersonName.Length - i - 1);

string str22 = new string(str2);

res = str11.ToString() + "." + str22.ToString();

if (res.Length > 20)

{

str1 = new char[20];

res.CopyTo(0, str1, 0, 20);

res = new string(str1);

}

res += "@domain.ua";

return res;

}

// Ðåøåíèå

string Name1 = "Andrey";

string Name2 = "Vasilkov";

string rez = Name1 + Name2 + "@domain.ua";

Console.WriteLine(rez);

//16)----------------------------------------------------------------------------------------------

foreach(string id in sourceIDs.Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))

{

sourceId = Convert.ToInt32(id);

break;

}

//Ðåøåíèå

foreach(string id in sourceIDs.Split(",", StringSplitOptions.RemoveEmptyEntries))

{

sourceId = Convert.ToInt32(id);

break;

}

//17)----------------------------------------------------------------------------------------------

string GetTextDiv2(string text)

{

int mid = text.Length / 2;

int r = text.IndexOf(" ", mid); if (r < 0) r = 5000;

int l = text.IndexOf(" ", 0, mid); if (l < 0) l = 5000;

if (r - mid > mid - l) // to left is closer

mid = l;

else mid = r;

if (mid == 5000) return "&nbsp" + text;

return "&nbsp" + text.Substring(0, mid) + " <br/>&nbsp" +

text.Substring(mid, text.Length - mid);

}

// Ðåøåíèå

string a = "aabbccdd";

string b = a.Substring(0, a.Length / 2);

string c = a.Substring(a.Length / 2, a.Length - a.Length / 2);

//18)----------------------------------------------------------------------------------------------

private static readonly char SPECIFIER = "$"[0];

private static readonly char DELIMITER = ":"[0];

private static readonly char[] DELIMITER\_ARRAY = new char[1]{ DELIMITER };

//19)----------------------------------------------------------------------------------------------

string mailTo = ((Config.GetSetting("AdminNotifications\_EmailAddress") == null) ||

(Config.GetSetting("AdminNotifications\_EmailAddress").Length <= 0)) ?

Globals.GetHostPortalSettings().HostSettings["SMTPPassword"].ToString() :

Config.GetSetting("AdminNotifications\_EmailAddress");

// Ðåøåíèå

string mailTo = ((Config.GetSetting("AdminNotifications\_EmailAddress") == null) ||

(Config.GetSetting("AdminNotifications\_EmailAddress").Length <= 0)) ?

Globals.GetHostPortalSettings().HostSettings["SMTPPassword"].ToString() :

Config.GetSetting("AdminNotifications\_EmailAddress"); // Áåç ïîíÿòèÿ ÷òî äåëàòü -\_'-'\_-

//20)----------------------------------------------------------------------------------------------

public bool CheckPath(string path)

{

int n;

n = 0;

//Ïðîâåðÿåì íàëè÷èå íóæíûõ ïàïîê;

if (Directory.Exists(path + "SCLAD"))

{

n += 1;

}

if (Directory.Exists(path + "REAL"))

{

n += 1;

}

if (Directory.Exists(path + "DOSTAVKA"))

{

n += 1;

}

//Ïðîâåðÿåì íàëè÷èå íóæíûõ ôàéëîâ

if (File.Exists(path + "analit.dbf"))

{

n += 1;

}

if (File.Exists(path + "partner.dbf"))

{

n += 1;

}

if (File.Exists(path + "SCLAD\\mdoc.dbf"))

{

n += 1;

}

if (File.Exists(path + "SCLAD\\mdoc.fpt"))

{

n += 1;

}

if (File.Exists(path + "SCLAD\\mdocm.dbf"))

{

n += 1;

}

if (File.Exists(path + "SCLAD\\mgrup.dbf"))

{

n += 1;

}

if (File.Exists(path + "SCLAD\\mlabel.dbf"))

{

n += 1;

}

if (File.Exists(path + "SCLAD\\mlabel.fpt"))

{

n += 1;

}

if (File.Exists(path + "REAL\\rbookm.dbf"))

{

n += 1;

}

if (File.Exists(path + "REAL\\rbook.dbf"))

{

n += 1;

}

if (File.Exists(path + "REAL\\rbook.fpt"))

{

n += 1;

}

if (File.Exists(path + "DOSTAVKA\\avt.dbf"))

{

n += 1;

}

if (File.Exists(path + "DOSTAVKA\\avtm.dbf"))

{

n += 1;

}

if (File.Exists(path + "DOSTAVKA\\avtm.fpt"))

{

n += 1;

}

if (File.Exists(path + "DOSTAVKA\\cargo.dbf"))

{

n += 1;

}

if (File.Exists(path + "DOSTAVKA\\cargom.dbf"))

{

n += 1;

}

if (File.Exists(path + "DOSTAVKA\\zamena.dbf"))

{

n += 1;

}

//Åñëè óêàçàííàÿ ïàïêà ñîäåðæèò âñå, ÷òî íóæíî

if (n == 20)

{

return true;

}

return false;

}

// Ðåøåíèå

static readonly string[] Dirs = new string[]{ "SCLAD", "REAL", "DOSTAVKA" };

static readonly string[] Files = new string[]{ "analit.dbf", "partner.dbf", "SCLAD\\mdoc.dbf" .. "DOSTAVKA\\zamena.dbf" };

public bool CheckPath(string path)

{

//Ïðîâåðÿåì íàëè÷èå íóæíûõ ïàïîê;

foreach(string s in Dirs)

if (!Directory.Exists(path + s)) return false;

//Ïðîâåðÿåì íàëè÷èå íóæíûõ ôàéëîâ

foreach(string s in Files)

if (!File.Exists(path + s)) return false;

return true;

}

//21)----------------------------------------------------------------------------------------------

txtContacts.Text = "";

bool first = true;

foreach(string contact in contacts)

{

if (first != true)

txtContacts.Text += ";";

first = false;

txtContacts.Text += contact;

}

// Ðåøåíèå

var result = from pl in contacts join t in txtContacts.Text on t equals pl select new { txtContacts.Text += contact };

// Âðîäå ïðî÷èòàë ÷òî ýòî ñàìîäåëêà join, íî êàê åãî ðåàëèçîâàòü íå çíàþ, êàê-òî òàê âðîäå.

//22)----------------------------------------------------------------------------------------------

if (Game1.clou == true)

{

Game1.clou = false;

}

else

{

Game1.clou = true;

}

// Ðåøåíèå

Game1.clou = !Game1.clou

© 2020 GitHub, Inc.

Terms

Privacy

Security

Status

Help

Contact GitHub

Pricing

API

Training

Blog

About