**Приложение А**

Листинг программы

private: System::Void timer1\_Tick(System::Object^ sender, System::EventArgs^ e) {

// Проверяем, достигли ли значение progressBar1 максимума

if (progressBar1->Value < progressBar1->Maximum) {

progressBar1->Value += progressBar1->Step; // Увеличиваем значение progressBar1 на заданный шаг

}

else {

timer1->Enabled = false; // Отключаем таймер, чтобы он больше не выполнял обновления

// Проверяем, существует ли уже открытая форма MyForm1

if (Application::OpenForms["MyForm1"] == nullptr) {

MyForm1^ newForm2 = gcnew MyForm1(); // Создаем указатель на новую форму MyForm1

newForm2->Show(); // Показываем новую форму

this->Hide(); // Скрываем текущую форму MyForm

}

}

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

MyForm2^ newForm3 = gcnew MyForm2(); //Создаем указатель на 3 форму

newForm3->MainForm1 = this; //Передаем 3 форме указатель на 2 форму

newForm3->Show();// Открываем новую форму

this->Hide(); // Опционально: скрываем текущую форму

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

SoundManager::ToggleSound();

try {

String^ iconPath = Path::Combine(

Application::StartupPath,

SoundManager::IsSoundEnabled ? "images\\music1.png" : "images\\music2.png"

);

if (File::Exists(iconPath)) {

button4->BackgroundImage = Image::FromFile(iconPath);

button4->BackgroundImageLayout = ImageLayout::Stretch; // Опционально: как растягивать изображение

}

else {

MessageBox::Show("Файл иконки звука не найден: " + iconPath,

"Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

catch (Exception^ ex) {

MessageBox::Show("Ошибка при обновлении иконки звука: " + ex->Message,

"Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

private: System::Void выходToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

private: System::Void справкаToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Help::ShowHelp(this, "5 Руководство пользователя.chm", HelpNavigator::Topic);

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

MyForm3^ newForm4 = gcnew MyForm3();

newForm4->MainForm2 = this;

newForm4->Show();

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

MyForm4^ newForm5 = gcnew MyForm4()

newForm5->MainForm3 = this;

newForm5->Show();

this->Hide();

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

MyForm5^ newForm6 = gcnew MyForm5();

newForm6->MainForm4 = this;

newForm6->Show();

this->Hide();

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

MyForm6^ newForm7 = gcnew MyForm6();

newForm7->MainForm5 = this;

newForm7->Show();

this->Hide();

}

private: System::Void выйтиИзИгрыToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

private: System::Void вернутсяНазадToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MainForm1->Show(); // Показываем главную форму

this->Close();// Опционально: скрываем текущую форму

}

void grid()

{

dataGridView1->Rows->Clear();

dataGridView1->Columns->Clear();

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

{

dataGridView1->Columns->Add(gcnew DataGridViewTextBoxColumn());

dataGridView1->Columns[i]->Width = 30;

dataGridView1->Columns[i]->SortMode = DataGridViewColumnSortMode::NotSortable;

}

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

{

dataGridView1->Rows->Add();

dataGridView1->Rows[i]->Height = 30;

}

}

static int sravDlin(String^ a, String^ b)

{

return b->Length - a->Length; // Сортировка по убыванию длины

}

void InitializeGame()

{

board = gcnew array<array<wchar\_t>^>(size);

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

{

board[i] = gcnew array<wchar\_t>(size);

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

{

board[i][j] = L' ';

}

}

words = gcnew List<Tuple<String^, int, int, int, int>^>();

sc = gcnew List<System::Drawing::Point>();

foundw = gcnew List<String^>();

s = false;

array<String^>^ wordList = {

"массив", "компилятор", "цикл", "интерпретатор",

"алгоритм", "функция", "параметр", "строка",

"компоновщик", "файл", "идентификатор", "компонент"

};

Array::Sort(wordList, gcnew Comparison<String^>(&MyForm3::sravDlin));

// Размещение слов на поле

for each (String ^ word in wordList)

{

if (!placeWord(word) && !placeWord(reverseString(word)))

{

// Если слово не удалось разместить ни в прямом, ни в обратном порядке

MessageBox::Show("Не удалось разместить слово: " + word,

"Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

Random^ rand = gcnew Random();

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

{

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

{

if (board[i][j] == L' ')

{

board[i][j] = L'а' + rand->Next(32);

}

}

}

// Заполнение ListBox словами для поиска

listBox1->Items->Clear();

for each (auto wordInfo in words)

{

listBox1->Items->Add(wordInfo->Item1);

}

upBoardDisplay();

label1->Text = "Найдено слов: 0/" + words->Count;

}

String^ reverseString(String^ s)

{

array<wchar\_t>^ chars = s->ToCharArray();

Array::Reverse(chars);

return gcnew String(chars);

}

bool placeWord(String^ word)

{

Random^ rand = gcnew Random();

int wordLength = word->Length;

for (int a = 0; a < 50; ++a)

{

int x = rand->Next(size - wordLength + 1); // Случайная строка (чтобы слово влезло по вертикали)

int y = rand->Next(size); // Случайный столбец

if (canplaceWord(word, x, y)) // Проверяем, можно ли разместить вертикально

{

doPlaceWord(word, x, y); // Размещаем вертикально

words->Add(Tuple::Create(word, x, y, x + wordLength - 1, y)); // Координаты для вертикального слова

return true;

}

}

return false;

}

bool canplaceWord(String^ word, int x, int y)

{

for (int i = 0; i < word->Length; i++)

{

if (x + i >= size) return false;

if (board[x + i][y] != L' ' && board[x + i][y] != word[i])

return false;

}

return true;

}

void doPlaceWord(String^ word, int x, int y)

{

for (int i = 0; i < word->Length; i++)

board[x + i][y] = word[i]; // Записываем буквы вертикально

}

void upBoardDisplay()

{

try

{

// Проверяем, что DataGridView инициализирован

if (dataGridView1 == nullptr || dataGridView1->Rows->Count != size || dataGridView1->Columns->Count != size)

{

grid(); // Переинициализируем сетку при несоответствии размеров

}

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

{

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

{

// Проверяем индексы

if (i >= dataGridView1->Rows->Count || j >= dataGridView1->Columns->Count)

{

continue; // Пропускаем несуществующие ячейки

}

// Получаем ячейку с проверкой

DataGridViewCell^ cell = dataGridView1->Rows[i]->Cells[j];

cell->Value = gcnew String(board[i][j].ToString());

// Всегда используем LightSalmon для текста

cell->Style->ForeColor = System::Drawing::Color::LightSalmon;

// Сброс остальных стилей

cell->Style->BackColor = dataGridView1->DefaultCellStyle->BackColor;

cell->Style->Font = gcnew System::Drawing::Font("PhantomMuff 1.5 Plus", 12,

System::Drawing::FontStyle::Regular);

// Проверка, найдено ли слово

bool found = false;

for each (auto wordInfo in words)

{

if (foundw->Contains(wordInfo->Item1))

{

int x1 = wordInfo->Item2, y1 = wordInfo->Item3;

int x2 = wordInfo->Item4, y2 = wordInfo->Item5;

if ((x1 == x2 && i == x1 && j >= y1 && j <= y2) ||

(y1 == y2 && j == y1 && i >= x1 && i <= x2))

{

found = true;

break;

}

}

}

// Стиль для найденных слов

if (found)

{

cell->Style->BackColor = System::Drawing::Color::LightGreen;

cell->Style->Font = gcnew System::Drawing::Font("PhantomMuff 1.5 Plus", 12,

System::Drawing::FontStyle::Bold);

}

// Стиль для выделенных ячеек

else if (sc->Contains(System::Drawing::Point(i, j)))

{

cell->Style->BackColor = Color::FromArgb(50, Color::Yellow);

cell->Style->Font = gcnew System::Drawing::Font("PhantomMuff 1.5 Plus", 14,

System::Drawing::FontStyle::Bold);

}

}

}

}

catch (Exception^ e)

{

MessageBox::Show("Ошибка при обновлении отображения: " + e->Message);

}

}

void upSelection(System::Drawing::Point start, System::Drawing::Point end)

{

sc->Clear();

// Вертикальное выделение (если столбцы совпадают)

if (start.Y == end.Y)

{

int x1 = Math::Min(start.X, end.X);

int x2 = Math::Max(start.X, end.X);

for (int x = x1; x <= x2; x++) {

sc->Add(System::Drawing::Point(x, start.Y));

}

}

// Горизонтальное выделение (если строки совпадают)

else if (start.X == end.X)

{

int y1 = Math::Min(start.Y, end.Y);

int y2 = Math::Max(start.Y, end.Y);

for (int y = y1; y <= y2; y++) {

sc->Add(System::Drawing::Point(start.X, y));

}

}

upBoardDisplay();

}

void checkSelection()

{

if (sc->Count < 2)

{

sc->Clear();

upBoardDisplay();

return;

}

String^ sw = "";

for each (System::Drawing::Point p in sc)

{

sw += board[p.X][p.Y];

}

bool wordFound = false;

for each (auto wordInfo in words)

{

if (!foundw->Contains(wordInfo->Item1))

{

// Проверка как прямого, так и обратного направления

if (sw->Equals(wordInfo->Item1, StringComparison::CurrentCultureIgnoreCase) ||

sw->Equals(reverseString(wordInfo->Item1), StringComparison::CurrentCultureIgnoreCase))

{

foundw->Add(wordInfo->Item1);

wordFound = true;

break;

}

}

}

if (wordFound)

{

label1->Text = "Найдено слов: " + foundw->Count + "/" + words->Count;

for (int i = 0; i < listBox1->Items->Count; i++)

{

String^ listItem = listBox1->Items[i]->ToString();

if (listItem->Equals(sw, StringComparison::CurrentCultureIgnoreCase) ||

listItem->Equals(reverseString(sw), StringComparison::CurrentCultureIgnoreCase))

{

for each (auto wi in words)

{

if (wi->Item1->Equals(sw, StringComparison::CurrentCultureIgnoreCase) ||

wi->Item1->Equals(reverseString(sw), StringComparison::CurrentCultureIgnoreCase))

{

listBox1->Items[i] = wi->Item1 + " +";

break;

}

}

break;

}

}

// Вставленный блок кода для проверки завершения игры

if (foundw->Count == words->Count)

{

System::Windows::Forms::DialogResult result = MessageBox::Show(

"Поздравляем! Вы нашли все слова!\n\nХотите сыграть еще?",

"Победа!",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes)

{

button1\_Click(nullptr, nullptr);

}

else

{

вернутсяНазадToolStripMenuItem\_Click(nullptr, nullptr);

}

}

}

sc->Clear();

upBoardDisplay();

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

grid();

InitializeGame();

}

private: System::Void dataGridView1\_CellMouseDown(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (e->RowIndex >= 0 && e->ColumnIndex >= 0)

{

sstart = System::Drawing::Point(e->RowIndex, e->ColumnIndex);

s = true;

sc->Clear();

sc->Add(sstart);

upBoardDisplay();

}

}

private: System::Void dataGridView1\_CellMouseMove(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (s && e->RowIndex >= 0 && e->ColumnIndex >= 0)

{

Point currentCell = Point(e->RowIndex, e->ColumnIndex);

if (sstart.X == currentCell.X || sstart.Y == currentCell.Y){

upSelection(sstart, currentCell);

}

}

}

private: System::Void dataGridView1\_CellMouseUp(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (s && e->RowIndex >= 0 && e->ColumnIndex >= 0)

{

checkSelection();

}

s = false;

}

private: System::Void выйтиИзИгрыToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

private: System::Void вернутсяНазадToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MainForm2->Show(); // Показываем главную форму

this->Close();// Опционально: скрываем текущую форму

}static void Initialize() {

Initialize("music\\Featherfall.wav");

}

// Основной метод инициализации

static void Initialize(System::String^ musicPath) {

if (player == nullptr) {

player = gcnew System::Media::SoundPlayer();

player->SoundLocation = System::IO::Path::Combine(

System::Windows::Forms::Application::StartupPath,

musicPath

);

try {

player->Load();

}

catch (...) {} // Проигнорировать ошибки загрузки

}

}

// Остальные методы без изменений

static void PlayMusic() {

if (IsSoundEnabled && player != nullptr) {

try {

player->PlayLooping();

}

catch (...) {}

}

}

static void StopMusic() {

if (player != nullptr) {

try {

player->Stop();

}

catch (...) {}

}

}

static void ToggleSound() {

IsSoundEnabled = !IsSoundEnabled;

if (IsSoundEnabled) PlayMusic();

else StopMusic();

}