Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

**ОТЧЕТ**

по учебной практике

Вариант 12

Студент гр. 351005 Галуха П.А.

Руководитель Данилова Г.В.

Минск 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc136223397)

[1 Постановка задачи 4](#_Toc136223398)

[2 Проектирование программного средства 5](#_Toc136223399)

[2.1 Структура программы 5](#_Toc136223400)

[2.2 Проектирование интерфейса программного средства 6](#_Toc136223401)

[2.3 Проектирование функционала программного средства 8](#_Toc136223402)

[3 Разработка программного средства 13](#_Toc136223403)

[3.1 Взаимодействие со списками 13](#_Toc136223407)

[3.2 Взаимодействие с таблицей 14](#_Toc136223408)

[3.3 Работа с файлами 15](#_Toc136223409)

[4 Тестирование программного средства 17](#_Toc136223410)

[5 Руководство пользователя 18](#_Toc136223411)

[5.1 Интерфейс программного средства 18](#_Toc136223412)

[5.2 Управление программным средством 21](#_Toc136223413)

[Заключение 22](#_Toc136223414)

[Список использованных источников 23](#_Toc136223415)

[Приложение A. Текст программы 24](#_Toc136223416)

# ВВЕДЕНИЕ

Память является ценным ресурсом в вычислительных системах, и программисту необходимо эффективно управлять ею. Для этого существуют два основных подхода к распределению памяти: статическое и динамическое.

Статическое распределение памяти происходит на этапе компиляции или трансляции программы. Здесь память заранее выделяется для переменных и данных, и они остаются неизменными в течение выполнения программы. Это позволяет достичь высокой производительности, поскольку нет необходимости в дополнительных операциях управления памятью во время выполнения программы. Однако этот подход требует заранее известного объема памяти, что может быть недостаточно гибким для некоторых задач.

В отличие от статического подхода, динамическое управление памятью осуществляется во время выполнения программы. Здесь память выделяется и освобождается по мере необходимости. Это позволяет более гибко использовать память и адаптироваться к изменяющимся требованиям программы. Например, когда требуется работа с переменным объемом данных или при создании структур данных, которые могут изменяться в размере.

Для организации динамической памяти используются указатели или ссылочные типы данных. Указатель представляет собой переменную, которая содержит адрес области памяти, где хранятся данные определенного типа. Указатели могут быть типизированными, то есть связанными с определенным типом данных, чтобы обеспечить безопасность и правильное обращение к данным. Область памяти для хранения указателей выделяется статически, в то время как память для переменных, на которые указывают указатели, выделяется динамически по мере необходимости.

Динамическое распределение памяти широко применяется в различных областях, включая обработку графики, звука, баз данных и других задач, где требуется гибкое использование памяти во время выполнения программы. Однако при использовании динамической памяти программисту также необходимо следить за освобождением уже ненужных участков памяти, чтобы избежать утечек памяти и эффективно использовать ресурсы системы.

Целью данной учебной практики является разработка программного средства “Избирательный архив” с использованием динамических структур.

# ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать приложение, которое позволит хранить и управлять информацией о кандидатах в предвыборной компании.

Запись списка кандидатов содержит следующие поля:

* номер округа;
* ФИО кандидата;
* партия;
* возраст;
* профессия.

Функциональные требования:

* формирование информационного бюллетеня;
* поиск полной информации по каждой партии;
* подсистема добавления, удаления и корректирования списка;
* подсистема отображения списка целиком.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 11.3 Community Edition.

**2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

**2.1 Структура программы**

При разработке приложения будет использовано одиннадцать модулей:

* MenuUnit – модуль, отвечающий за отображение главного меню приложения. Он предоставляет пользователю список доступных функций и действий, которые можно выполнить в приложении.
* ViewCandidateListUnit – модуль, ответственный за отображение окна со списком кандидатов. Он позволяет пользователю просматривать информацию о кандидатах.
* PartyUnit – модуль, отображающий информацию о партиях в соответствии с определенными критериями. Он предоставляет пользователю возможность просматривать информацию о различных политических партиях.
* SearchUnit – модуль, обеспечивающий поиск полной информации о конкретной партии. Он позволяет пользователю искать и получать подробную информацию о партии.
* ModifyCandidateUnit – модуль, позволяющий пользователю добавлять новые записи в список кандидатов. Он обеспечивает возможность ввода данных о новых кандидатах.
* CandidateListUnit – модуль, содержащий реализацию однонаправленного списка для работы с кандидатами. Он предоставляет функциональность для управления списком кандидатов.
* PartyListUnit – модуль, содержащий реализацию однонаправленного списка для работы с партиями. Он предоставляет функциональность для управления списком партий.
* ProfessionListUnit – модуль, содержащий реализацию однонаправленного списка для работы с профессиями. Он предоставляет функциональность для управления списком профессий.
* FileUnit – модуль, обеспечивающий функциональность работы с файлами. Он позволяет пользователю открывать и сохранять файлы, связанные с кандидатами, партиями или другими данными приложения.
* PositiveNumberComponentUnit – модуль, который обеспечивает корректный ввод положительных чисел пользователем. Он проверяет введенные значения на соответствие требованиям положительных чисел и предоставляет возможность ввода только допустимых значений.
* NameComponentUnit – модуль, обеспечивающий корректный ввод букв алфавита пользователем. Он проверяет введенные значения на соответствие требованиям алфавита и предоставляет возможность ввода только допустимых буквенных символов.

**2.2 Проектирование интерфейса программного средства**

Внешний вид программных средств играет существенную роль в обеспечении удобства пользователей и качества самого средства.

Интерфейс должен удовлетворять следующим требованиям:

* простота и понятность в использовании;
* удобное расположение функций.

**2.2.1** Главное окно

Главное окно приложения состоит из пяти кнопок, с помощью которых пользователь может переходить между окнами программного средства:

* первая кнопка открывает модуль “ViewCandidateListUnit”, где пользователь может просмотреть список кандидатов;
* вторая кнопка позволяет перейти в модуль “PartyUnit”, где предоставлена информация о партиях;
* третья кнопка, названная "Поиск партии", позволяет пользователю ввести название партии и просмотреть список кандидатов в этой партии;
* четвертая кнопка отображает справку о программном средстве;
* пятая кнопка предоставляет информацию о разработчике программного средства.

Макет главного окна приложения представлен на рисунке 2.1.

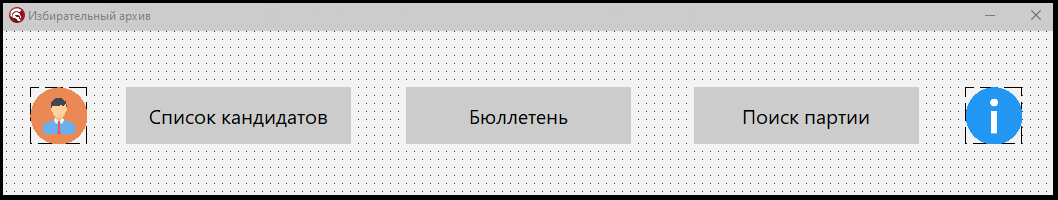


Рисунок 2.1 – Макет главного меню приложения

**2.2.2** Окно просмотра списка

Окно просмотра списка состоит из кнопок, информационного поля и списка. В верхней части формы находится строка с кнопками:

* добавить новую запись;
* изменить выбранную запись;
* удалить выбранную запись.

Отображение списка кандидатов осуществляются с помощью компонента TStringGrid. В таблице для кандидатов будут присутствовать такие поля как:

* номер округа;
* ФИО кандидата;
* партия;
* возраст;
* профессия.

Макет окна просмотра списка представлен на рисунке 2.2.

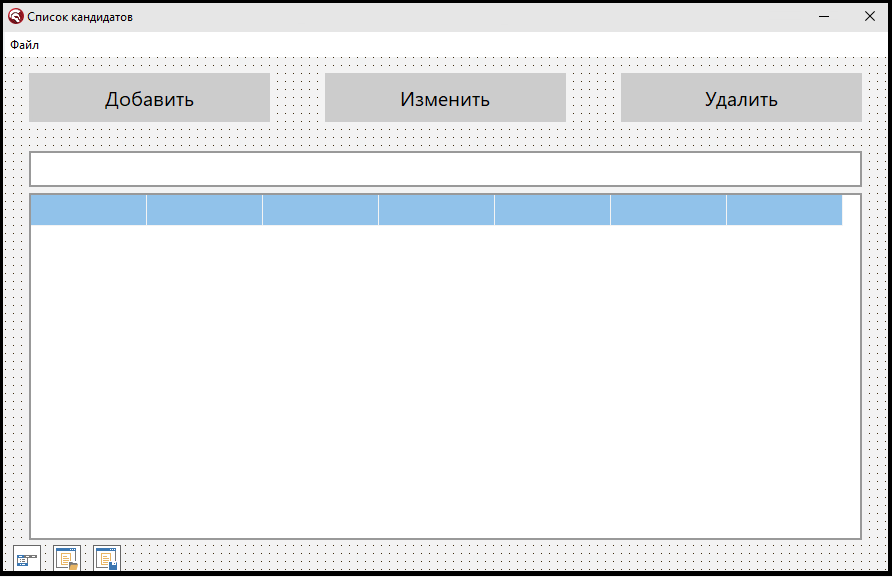


Рисунок 2.2 – Макет окна просмотра списка

**2.2.3** Окно добавления кандидата

Окно добавления кандидата состоит из нескольких однострочных полей ввода и двух кнопок. Выбор подтверждается нажатием на компонент TButton. Макет окна добавления кандидата представлен на рисунке 2.3.

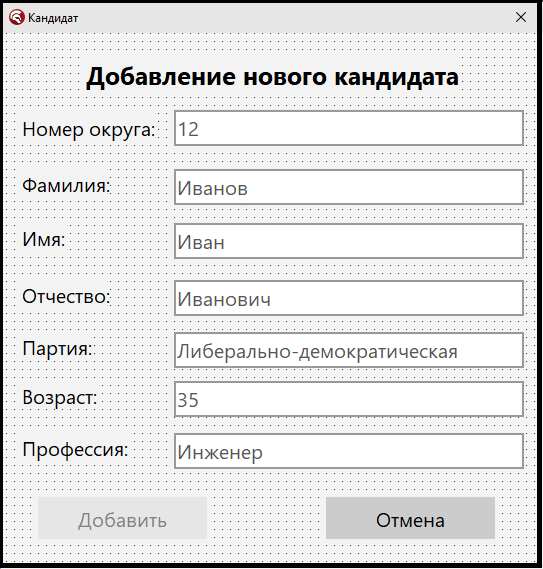


Рисунок 2.3 – Макет окна добавления кандидата

**2.2.4** Окно формирования бюллетеня

Окно формирования бюллетеня состоит из информационного поля и таблицы. Макет окна формирования бюллетеня представлен на рисунке 2.4.

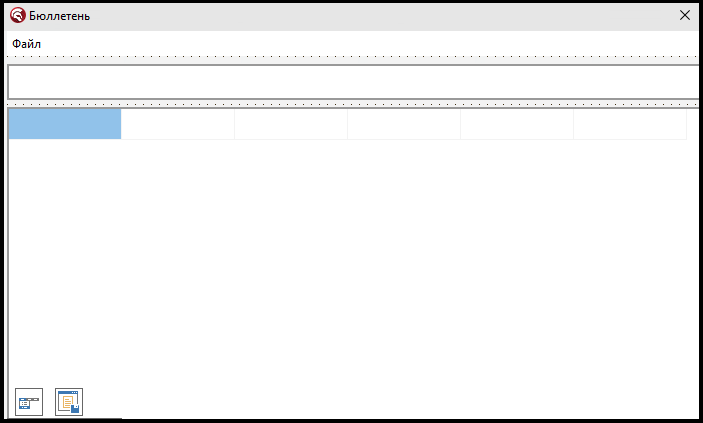


Рисунок 2.4 – Макет окна формирования бюллетеня

**2.2.5** Окно поиска

Окно поиска состоит из однострочного поля для ввода названия проекта, а также кнопки для поиска. Макет окна поиска представлен на рисунке 2.5.

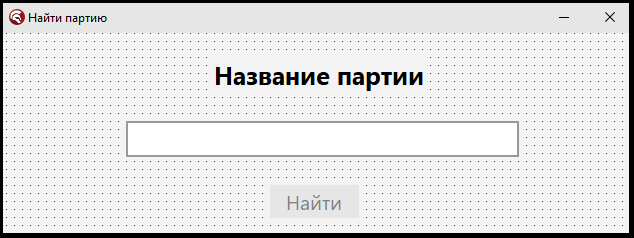


Рисунок 2.5 – Макет окна поиска

**2.3 Проектирование функционала программного средства**

Для работы программы был выбран тип данных “однонаправленный список”.

Для решения поставленной задачи потребуется разработать несколько алгоритмов, реализующих основные функции программы:

* поиск полной информации по каждой партии;
* формирование информационного бюллетеня;
* добавление записи о кандидате;
* сохранение списка в файл.

**2.3.1** Поиск полной информации по каждой партии

Для получения полной информации о каждой партии необходимо просмотреть список партий от начала до конца и сравнивать название партии каждого кандидата с искомым. Блок-схема процедуры нахождения списка всех кандидатов партии по названию.



Рисунок 2.6 – Блок-схема процедуры FillPartyStringGrid

**2.3.2** Формирование информационного бюллетеня

Для формирования информационного бюллетеня первоначально создается пустой список. Затем он наполняется партиями, используя функцию-фильтр, которая добавляет или изменяет количество партий в зависимости от того, добавлены ли уже эти партии. Блок-схема процедуры для формирования информационного бюллетеня представлена ниже.



Рисунок 2.7 – Блок-схема процедуры AddParty

**2.3.3** Добавление записи в список

Для добавления записи о кандидате будет использоваться процедура, которая будет создавать новую запись о кандидате, основываясь на переданном ей параметре, и помещать запись в конец списка. Блок-схема процедуры для добавления кандидате представлена ниже.



Рисунок 2.8 – Блок-схема процедуры AddCandidate

**2.3.4** Сохранение списка

Процедура сохранения списка кандидатов записывает данные о каждом кандидате в типизированный файл до того момента, пока указатель на следующую запись будет задан определенным значением. Блок-схема сохранения списка кандидатов представлена ниже.



Рисунок 2.9 – Блок-схема процедуры WriteFileData

Если файл с именем, переданным в качестве параметра в процедуру, не был найден, то данная процедура создает его и производит запись.

# РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА



## **Взаимодействие со списками**

Для хранения информации о кандидатах будет использоваться динамическая структура данных однонаправленный список.

**3.1.1** Добавление записи в список

Для добавления записи о кандидате будет использоваться процедура, которая будет создавать новую запись о кандидате, основываясь на переданном ей параметре, и помещать запись в конец списка. Код процедуры для добавления кандидата представлен ниже.

Procedure AddCandidate(Var CandidateList: PCandidate; Const CandidateData:

TCandidateData);

Var

NewCandidate, CurrCandidate: PCandidate;

Begin

// Создание нового кандидата

NewCandidate := CreateCandidate(CandidateData);

// Если до этого список был пуст

If CandidateList = Nil Then

CandidateList := NewCandidate

// Если до этого список имел элементы

Else

Begin

// Перемещение указателя в конец списка

CurrCandidate := CandidateList;

While CurrCandidate^.Next <> Nil Do

CurrCandidate := CurrCandidate^.Next;

// Добавление нового кандидата в конец списка

CurrCandidate^.Next := NewCandidate;

End;

End;

**3.1.2** Удаление списка

Для очистки списка будет использоваться процедура ClearCandidates. Код процедуры для удаления списка представлен ниже.

Procedure ClearCandidates(Var CandidateList: PCandidate);

Var

CurrCandidate, TempCandidate: PCandidate;

Begin

// Получение указателя на начало списка

CurrCandidate := CandidateList;

// Проход по всему списку

While CurrCandidate <> Nil Do

Begin

// Сохранение удаляемого кандидата

TempCandidate := CurrCandidate;

// Создание новой связи

CurrCandidate := CurrCandidate^.Next;

// Очищение памяти

Dispose(TempCandidate);

End;

// Разрывание связи указателя списка

CandidateList := Nil;

End;

## Взаимодействие с таблицей

Взаимодействие с таблицей производится после вызова модального окна и изменения списка кандидатов.

**3.2.1** Добавление новой строки

Для создания новой записи о кандидате необходимо использовать метод AddButtonClick, который будет вызывать открытие специальной формы для заполнения данных. Код метода для добавления новой строки в таблицу представлен ниже.

Procedure TViewCandidateListForm.AddButtonClick(Sender: TObject);

Begin

// Добавлено ли предельное количество кандидатов

If CountCandidates(CandidateList) < MAX\_CANDIDATE\_COUNT Then

Begin

// Установление типа формы

StateForm := Add;

// Создание формы добавления

CreateModifyForm('Добавление кандидата', 'Добавить');

// Нажата кнопка добавить

If IsModified Then

Begin

// Увеличение числа строк на 1

CandidateListStringGrid.RowCount := CandidateListStringGrid.RowCount +

1;

// Заполнение таблицы

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

IsModified := False;

// Изменение параметров сохранения

IsSaved := False;

SaveMenuItem.Enabled := True;

// Очищение информационнго поля

ContentEdit.Text := '';

End;

End

Else

// Сообщение об ошибке

Application.MessageBox(PWideChar('Количество кандидатов не может превышать '

+ IntToStr(MAX\_CANDIDATE\_COUNT) + ' !'),

'Предупреждение', MB\_OK + MB\_ICONWARNING);

End;

**3.2.2** Удаление строки

Для удаления информации о кандидате необходимо использовать метод DeleteButtonClick. Код метода для удаления строки в таблице представлен ниже.

Procedure TViewCandidateListForm.DeleteButtonClick(Sender: TObject);

Var

Confirmation, CandidateCount: Integer;

Begin

// Выбран редактируемый ряд

If CandidateListStringGrid.Row > 0 Then

Begin

// Предупреждение о удалении

Confirmation := Application.MessageBox('Вы действительно хотите удалить

кандидата?', 'Удаление кандидата',

MB\_YESNO + MB\_ICONQUESTION +

MB\_DEFBUTTON2);

If Confirmation = IDYES Then

Begin

// Удаление кандидата

DeleteCandidate(CandidateList,

GetCandidateData(CandidateListStringGrid));

// Уменьшение числа строк на 1

CandidateListStringGrid.RowCount := CandidateListStringGrid.RowCount –

1;

// Заполнение таблицы

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

// Подсчет числа кандидатов

CandidateCount := CountCandidates(CandidateList);

// Изменение параметров сохранения

IsSaved := CandidateCount = 0;

SaveMenuItem.Enabled := CandidateCount > 0;

// Очищение информационнго поля

ContentEdit.Text := '';

End;

End

Else

// Сообщение об ошибке

Application.MessageBox('Не выбрано редактируемое поле!', 'Предупреждение',

MB\_OK + MB\_ICONWARNING);

End;

## **Работа с файлами**

В данной программе, как и было упомянуто ранее, предусмотрена пользовательская работа с файлами, а именно, загрузка и сохранение данных. Функция загрузки файлов позволяет пользователю выбрать файл на своем компьютере и загрузить его в программу.

**3.3.1** Сохранение списка в файл

Для сохранения списка в файл используется метод SaveMenuItemClick.

Код метода SaveMenuItemClick представлен ниже.

Procedure TViewCandidateListForm.SaveMenuItemClick(Sender: TObject);

Var

OutputFile: TCandidateDataFile;

Begin

// Диалоговое окно сохранения файла

If SaveTextFileDialog.Execute Then

Begin

AssignFile(OutputFile, SaveTextFileDialog.FileName);

// Вызов функции записи в файл

WriteFileData(OutputFile, CandidateList);

// Результат сохранен

IsSaved := True;

End;

End;

**3.3.2** Добавление записи в список из файла

Для добавления записи в список из файла используется процедура OpenMenuItemClick. Код процедуры OpenMenuItemClick представлен ниже.

Procedure TViewCandidateListForm.OpenMenuItemClick(Sender: TObject);

Var

InputFile: TCandidateDataFile;

TempCandidateList: PCandidate;

IsCorrect: Boolean;

Begin

// Диалоговое окно открытия файла

If OpenTextFileDialog.Execute Then

Begin

AssignFile(InputFile, OpenTextFileDialog.FileName);

// Инициализация списка

TempCandidateList := Nil;

// Чтение данных из файла

IsCorrect := ReadFileData(InputFile, TempCandidateList);

// Если успешное чтение

If IsCorrect Then

Begin

// Очищение старых данных

ClearCandidates(CandidateList);

// Запись новых данных

RecordFileData(CandidateList,TempCandidateList,CandidateListStringGrid);

// Изменение параметров сохранения

IsSaved := True;

SaveMenuItem.Enabled := True;

End

Else

Begin

// Очищение новых данных

ClearCandidates(TempCandidateList);

// Сообщение об ошибке

Application.MessageBox('Содержимое файла повреждено!', 'Ошибка', MB\_OK +

MB\_ICONERROR);

End;

End;

End;

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Основной проблемой было некорректное формирование бюллетеня. При формировании возникала ошибка неверного доступа к памяти. Это было связано с тем, что не инициализировалось поле, указывающее на популярную профессию. И при формировании бюллетеня первый элемент содержал некорректные данные. Проблема представлена на рисунке 4.1.

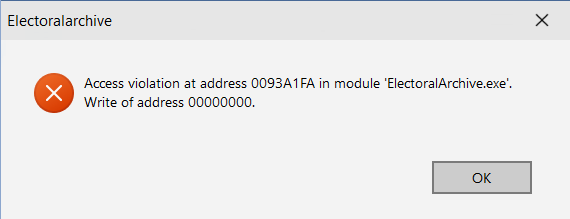


Рисунок 4.1 – Некорректное формирование бюллетеня

Проблема была решена изменением функции создания нового узла, которая имеет вид:

Function CreateParty(CandidateData: TCandidateData) : PParty;

Var

NewParty: PParty;

Begin

New(NewParty);

NewParty^.Data.PartyName := CandidateData.Party;

NewParty^.Data.ApplicationCount := 1;

NewParty^.Data.Age := CandidateData.Age;

NewParty^.Data.ProfessionList := Nil;

AddProfession(NewParty^.Data.ProfessionList, CandidateData.Profession);

NewParty^.Next := Nil;

CreateParty := NewParty;

End;

Большинство проблем возникло из-за недочетов на стадии проектирования программного средства, на стадии тестирования приложения все проблемы были исправлены.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

**5.1 Интерфейс программного средства**

**5.1.1** Главное окно

Главное меню приложения состоит из трех основных и двух дополнительных кнопок:

* Первая располагается кнопка “Список кандидатов”, по нажатию которой пользователь переходит в модуль ViewCandidateListUnit, в котором он может просмотреть список существующих кандидатов, добавить новых, удалить или редактировать старых.
* Вторая кнопка позволяет просмотреть список партий и их краткие характеристики. Данная кнопка отображает результат после того, как в списке кандидатов будет добавлена как минимум одна запись.
* Третья располагается “Поиск партии”, нажав на которую, пользователь может ввести название партии и просмотреть список кандидатов по партии. Данная кнопка отображает результат после того, как в списке кандидатов будет добавлена как минимум одна запись.
* Четвертая кнопка выводит справку о программном средстве.
* Пятая располагается кнопка для получения информации о разработчике программного средства.

Внешний вид главного меню приложения представлен на рисунке 5.1.

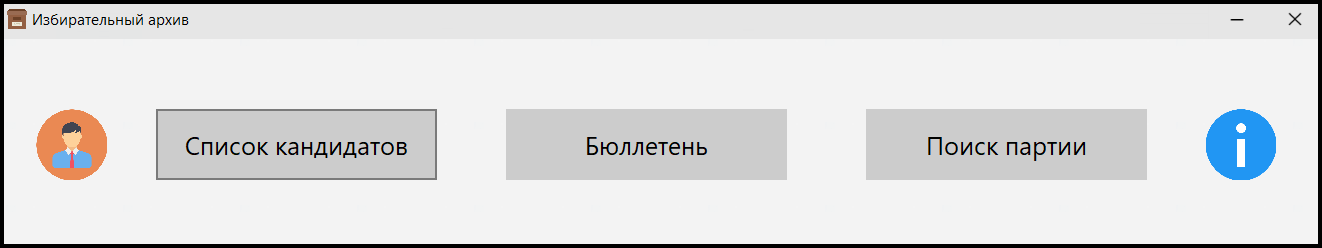


Рисунок 5.1 – Главное меню приложения

**5.1.2** Окно просмотра списка

В верхней части окна находится строка меню, где пользователь может загрузить в таблицу данные с файла или сохранить данные в файл. Данная кнопка становится неактивной, если в таблице нет записей. Ниже располагается строка с кнопками для редактирования, удаления или добавления новой записи о сотруднике или задаче сотрудника. Внешний вид окна просмотра списка представлен на рисунке 5.2

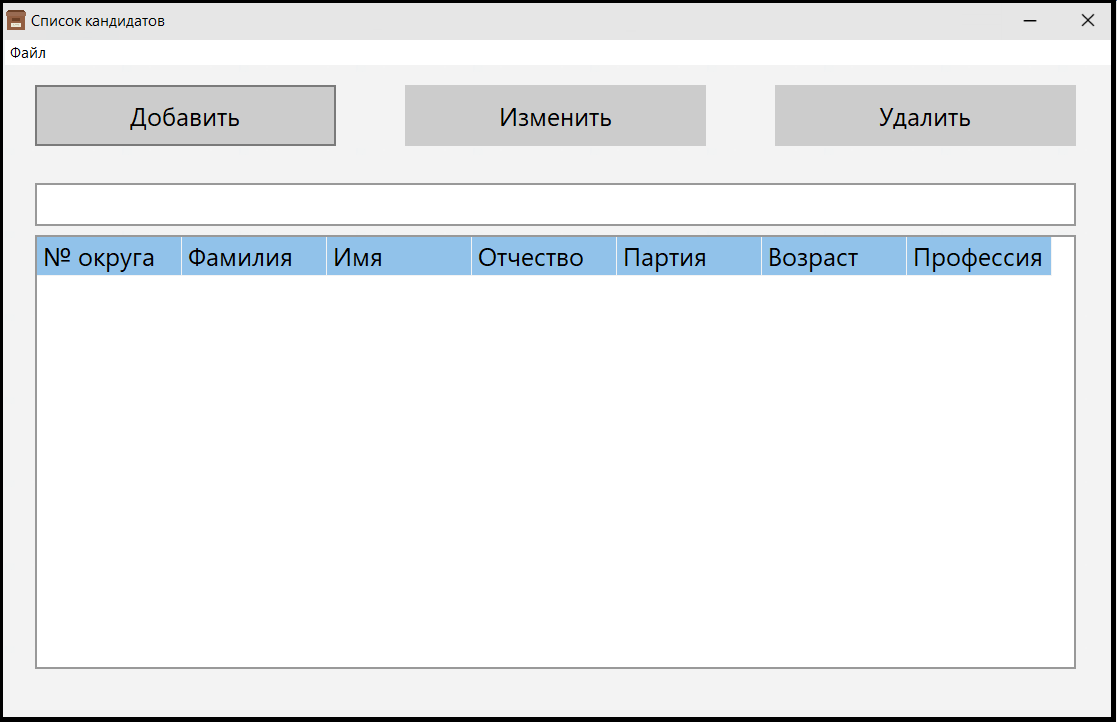


Рисунок 5.2 – Окно просмотра списка

**5.1.3** Окно добавления кандидата

Окно добавления кандидата состоит из текстовых полей, где пользователь должен добавить информацию о кандидате. Полная информация должна быть уникальной. Кнопка “Добавить” станет активной после корректного заполнения всех полей. Внешний вид окна представлен на рисунке 5.3.

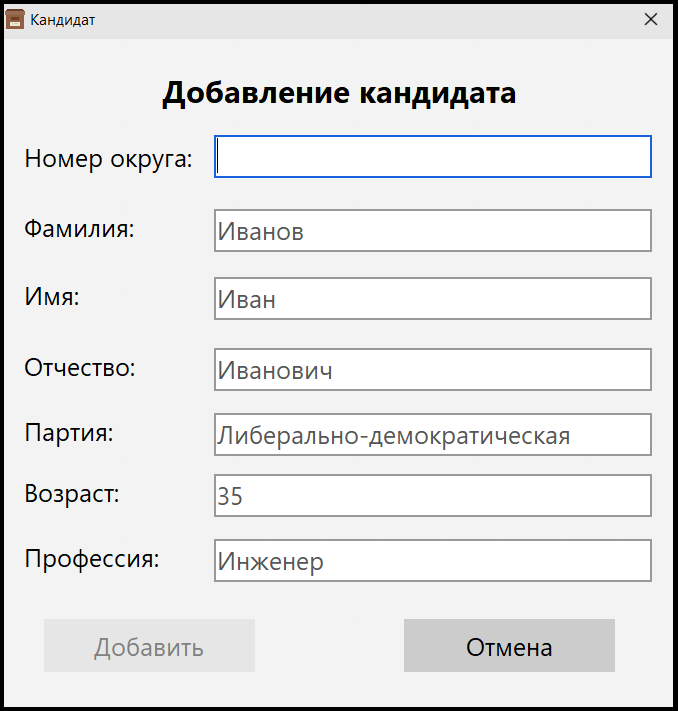


Рисунок 5.3 – Окно добавления кандидата

**5.1.4** Окно информационного бюллетеня

Окно информационного бюллетеня состоит из информационного поля и таблицы. Для получения полной информации о конкретном поле, вам следует щелкнуть на соответствующей ячейке. Внешний вид окна представлен на рисунке 5.4.

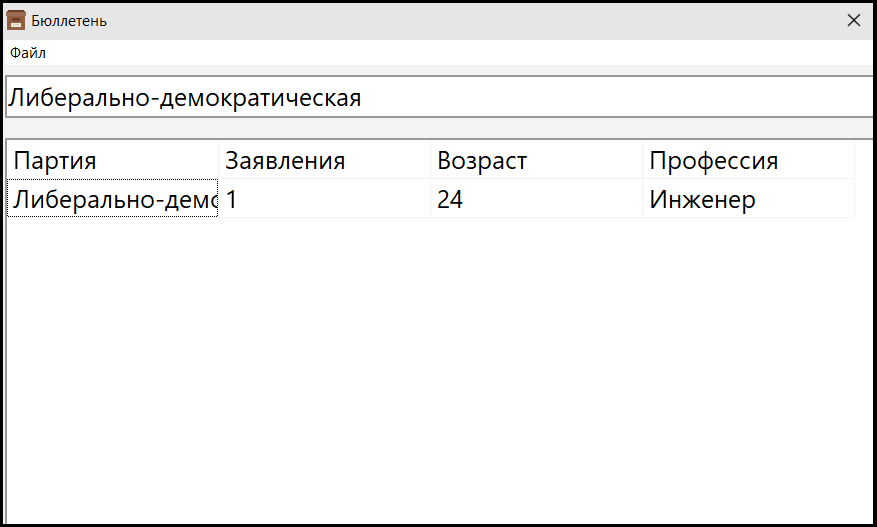


Рисунок 5.4 – Окно информационного бюллетеня

**5.1.5** Окно поиска полной информации о партии

Окно поиска состоит из однострочного поля ввода, предназначенного для названия партии и кнопоки “Найти”. Кнопка становится активной только после корректного ввода данных. Внешний вид окна поиска полной информации о партии представлен на рисунке 5.5.

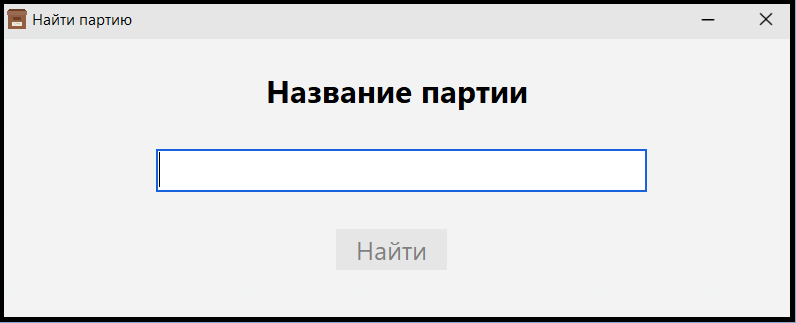


Рисунок 5.5 – Окно поиска полной информации о партии

**5.2 Управление программным средством**

Для начала работы нужно добавить как минимум одного кандидата. После добавления записи в список кандидатов активными станут кнопки “Бюллетень” и “Поиск партии”.

Для перехода к просмотру списка, добавления записи, удаления записи, редактирования записи нужно нажать кнопку “Список кандидатов”.

Для ввода персональных данных в программу используется два способа: ввод данных через пользовательский интерфейс, импорт данных из файла.

Ввод данных через пользовательский интерфейс осуществляется в специальном окне, где пользователь вносит персональные данные кандидата в отведенные текстовые поля.

Выходными данными является полученная во время выполнения программы информация о кандидатах в рамках конкретной партии, а также список партий.

Вывод данных также осуществляется в двух форматах: в виде типизированного файла в конкретном формате и списками, отображающимися в виде таблиц для прямого редактирования. Внешний вид окна списка с таблицами для редактрования представлен на рисунке 5.6.

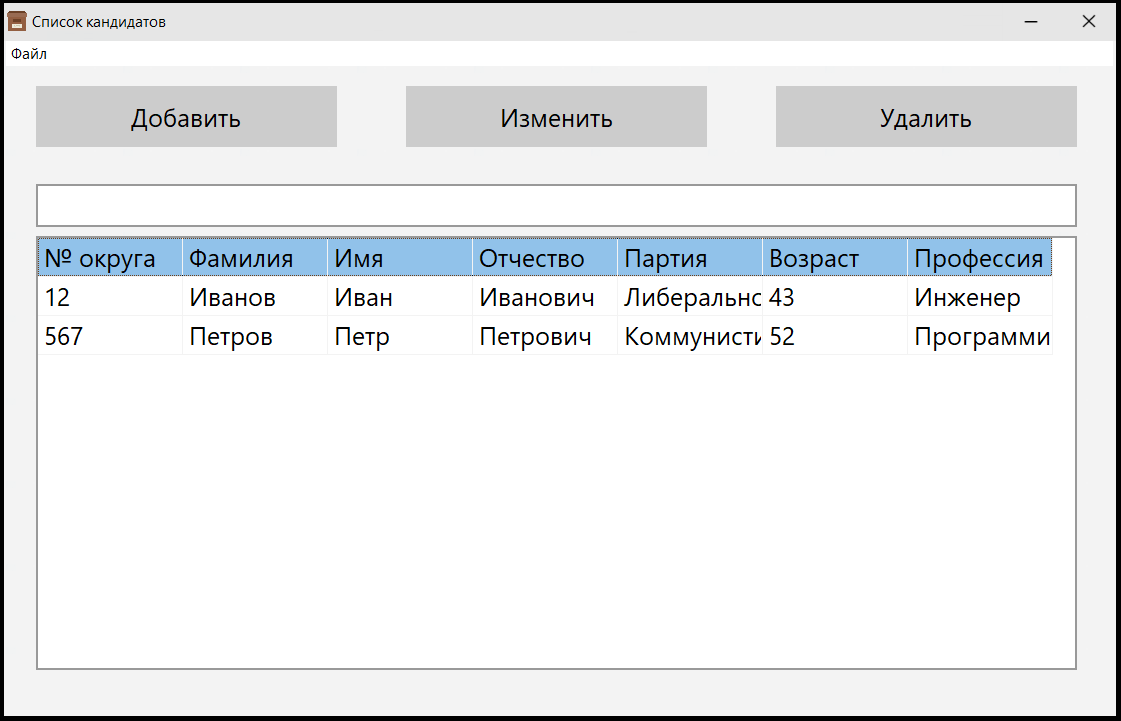


Рисунок 5.6 – Внешний вид окна списка

**ЗАКЛЮЧЕНИЕ**

В настоящее время программные средства, позволяющие взаимодействовать со списком кандидатов, пользуются большой популярностью. В условиях современной политики, где эффективное распределение времени играет важную роль, такие инструменты становятся неотъемлемой частью рабочего процесса. Предвыборные компании стремятся использовать современные программные решения, которые упрощают и автоматизируют управление, улучшают коммуникацию и способствуют достижению поставленных целей.

В рамках данной учебной практики было разработано программное средство для предвыборной компании, которое обеспечивает возможность взаимодействия со списком кандидатов. Программное средство позволяет пользователю создавать и управлять списком кандидатов, формировать бюллетень, искать партии. При разработке программы были успешно выполнены все поставленные задачи, а именно:

* формирование информационного бюллетеня;
* осуществление поиска полной информации по каждой партии;
* подсистема добавления, удаления и корректирования списков;
* подсистема отображения списка целиком.

Для успешного достижения всех поставленных целей при разработке данного приложения было необходимо углубленно изучить объектно-ориентированные и формовые возможности языка программирования Delphi. Это позволило эффективно использовать функциональность данного языка при создании пользовательского интерфейса, обеспечить визуальное представление данных и взаимодействие с пользователем. Усвоение указанных аспектов языка позволило успешно реализовать все поставленные задачи в рамках разработки приложения.

В дальнейшем улучшении приложения планируется включить добавление функционала поиска нужного кандидата, а также улучшение графического интерфейса.

Данное приложение позволяет оптимизировать управление задачами и ресурсами, обеспечивает легкость использования и надежность, а также способствует достижению успеха в проведении предвыборной компании.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Григорьев А.Б. О чем не пишут в книгах по Delphi [Текст]. – СПб.: БХВ-Петербург, 2010. – 576 с. – ISBN 978-5-699-40703-3.
2. Парижский С.М. Delphi. Учимся на примерах /Под ред. Ю. А. Шпака [Текст]. – Киев: МК-Пресс, 2005. – 216 с. – ISBN 978-5-9775-0516-1.
3. Синицын С.В. Программирование на языке высокого уровня / [Текст]. – М.: Академия, 2010. – 400 с. – ISBN 978-5-7695-6673-8.
4. Фаронов В. Delphi. Программирование на языке высокого уровня [Текст]: Учебник для вузов. – СПб.: Питер, 2009. – 640 с. – ISBN 978-5-9775-0657-1.
5. Фаулер М. Предметно-ориентированные языки программирования / [Текст]. – М.: Вильямс, 2011. – 576 с. – ISBN 978-5-8459-1738-6.
6. Фленов М.Е. Библия Delphi [Текст]. 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008. – 800 с. – ISBN 978-5-397-01304-8.
7. Чиртик А. Программирование в Delphi. Трюки и эффекты [Текст]. – СПб.: Питер, 2011. – 400 с. – ISBN 978-5-8046-0008-3.
8. Шпак Ю.А. Delphi 7 на примерах [Текст]/Под ред. Ю.С. Ковтанюка. – Киев: Юниор, 2006. – 416 с. – ISBN 966-8806-22-0.

# 

# ПРИЛОЖЕНИЕ A

Текст программы

Unit MenuUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Imaging.pngimage,

Vcl.ExtCtrls,

ViewCandidateListUnit, PartyUnit, SearchUnit, CandidateListUnit;

Type

TStateForm = (Bulletin, Party, Add, Edit);

TMenuForm = Class(TForm)

DeveloperImage: TImage;

CandidateListButton: TButton;

BulletinButton: TButton;

SearchPartyButton: TButton;

InfoImage: TImage;

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

Procedure FormDestroy(Sender: TObject);

Procedure DeveloperImageClick(Sender: TObject);

Procedure InfoImageClick(Sender: TObject);

Procedure CandidateListButtonClick(Sender: TObject);

Procedure BulletinButtonClick(Sender: TObject);

Procedure SearchPartyButtonClick(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

MenuForm: TMenuForm;

CandidateList: PCandidate = Nil;

IsSaved: Boolean = True;

StateForm: TStateForm;

Implementation

{$R \*.dfm}

Function TMenuForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Begin

CallHelp := False;

FormHelp := False;

End;

Procedure TMenuForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Begin

If Key = VK\_F1 Then

InfoImageClick(InfoImage);

End;

Procedure TMenuForm.FormDestroy(Sender: TObject);

Begin

ClearCandidates(CandidateList);

End;

Procedure CreateModalForm(CaptionText, LabelText: String; ModalWidth, ModalHeight:

Integer);

Var

ModalForm: TForm;

ModalLabel: TLabel;

Begin

ModalForm := TForm.Create(Nil);

Try

ModalForm.BorderIcons := [BiSystemMenu];

ModalForm.BorderStyle := BsSingle;

ModalForm.Caption := CaptionText;

ModalForm.Height := ModalHeight;

ModalForm.Icon := MenuForm.Icon;

ModalForm.Position := PoScreenCenter;

ModalForm.Width := ModalWidth;

ModalForm.OnHelp := MenuForm.FormHelp;

ModalLabel := TLabel.Create(ModalForm);

ModalLabel.Caption := LabelText;

ModalLabel.Font.Size := 12;

ModalLabel.Left := (ModalForm.ClientWidth - ModalLabel.Width) Div 2;

ModalLabel.Parent := ModalForm;

ModalLabel.Top := (ModalForm.ClientHeight - ModalLabel.Height) Div 2;

ModalForm.ShowModal;

Finally

ModalForm.Free;

End;

End;

Procedure TMenuForm.DeveloperImageClick(Sender: TObject);

Begin

CreateModalForm('О разработчике', 'Группа: 351005'#13#10 +

'Разработчик: Галуха Павел

Александрович'#13#10 +

'Телеграмм: @pavello06', 500, 150);

End;

Procedure TMenuForm.InfoImageClick(Sender: TObject);

Begin

CreateModalForm('Инструкция',

'Список кандидатов:'#13#10 +

'1. Для добавления кандидата нажмите кнопку "Добавить" или

нажмите клавишу Ins.'#13#10 +

'2. Для изменения кандидата нажмите кнопку "Изменить" или

нажмите дважды на интересующий ряд.'#13#10 +

'3. Для удаления кандидата нажмите кнопку "Удалить" или нажмите

клавишу Del.'#13#10 +

'4. Для получения диапазона для ввода поля наведитесь на

него.'#13#10#13#10 +

'Файлы:'#13#10 +

'1. Файл имеет расширение .ea'#13#10#13#10 +

'Поиск:'#13#10 +

'Для использования поиска введите название партии и нажмите

кнопку "Найти".',

1000, 450);

End;

Procedure CreateForm(FormClass: TFormClass);

Var

Form: TForm;

Begin

MenuForm.Visible := False;

Form := FormClass.Create(MenuForm);

Form.Icon := MenuForm.Icon;

Form.ShowModal;

MenuForm.Visible := True;

End;

Procedure TMenuForm.CandidateListButtonClick(Sender: TObject);

Begin

CreateForm(TViewCandidateListForm);

End;

Procedure TMenuForm.BulletinButtonClick(Sender: TObject);

Begin

If CandidateList = Nil Then

Application.MessageBox('Вы еще не добавили кандидатов!', 'Предупреждение',

MB\_OK + MB\_ICONWARNING)

Else

Begin

StateForm := Bulletin;

CreateForm(TPartyForm);

End;

End;

Procedure TMenuForm.SearchPartyButtonClick(Sender: TObject);

Begin

If CandidateList = Nil Then

Application.MessageBox('Вы еще не добавили кандидатов!', 'Предупреждение',

MB\_OK + MB\_ICONWARNING)

Else

Begin

StateForm := Party;

CreateForm(TSearchForm);

End;

End;

Procedure TMenuForm.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

Var

Confirmation: Integer;

Begin

If IsSaved Then

Begin

Confirmation := Application.MessageBox('Вы действительно хотите выйти?',

'Выход', MB\_YESNO + MB\_ICONQUESTION +

MB\_DEFBUTTON2);

CanClose := Confirmation = IDYES;

End

Else

Begin

Confirmation := Application.MessageBox('Вы не сохранили файл, хотите ли

сохранить файл?', 'Выход',

MB\_YESNOCANCEl + MB\_ICONQUESTION +

MB\_DEFBUTTON2);

Case Confirmation Of

mrYes:

Begin

ViewCandidateListForm.SaveMenuItemClick(Sender);

CanClose := IsSaved;

End;

mrNo:

CanClose := True;

mrCancel:

CanClose := False;

End;

End;

End;

End.

Unit ViewCandidateListUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Vcl.StdCtrls, Vcl.Grids,

Vcl.ExtDlgs,

ModifyCandidateUnit, CandidateListUnit, FileUnit;

Type

TViewCandidateListForm = Class(TForm)

MainMenu: TMainMenu;

FileMenuItem: TMenuItem;

OpenMenuItem: TMenuItem;

SaveMenuItem: TMenuItem;

OpenTextFileDialog: TOpenTextFileDialog;

SaveTextFileDialog: TSaveTextFileDialog;

AddButton: TButton;

EditButton: TButton;

DeleteButton: TButton;

ContentEdit: TEdit;

CandidateListStringGrid: TStringGrid;

Procedure CreateParams(Var Params: TCreateParams); Override;

Procedure FormCreate(Sender: TObject);

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Procedure FormShow(Sender: TObject);

Procedure OpenMenuItemClick(Sender: TObject);

Procedure SaveMenuItemClick(Sender: TObject);

Procedure AddButtonClick(Sender: TObject);

Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure EditButtonClick(Sender: TObject);

Procedure CandidateListStringGridDblClick(Sender: TObject);

Procedure DeleteButtonClick(Sender: TObject);

Procedure CandidateListStringGridKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure CandidateListStringGridSelectCell(Sender: TObject; ACol, ARow:

Integer; Var CanSelect:

Boolean);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

ViewCandidateListForm: TViewCandidateListForm;

OldCandidateData: TCandidateData;

IsModified: Boolean = False;

Implementation

{$R \*.dfm}

Uses

MenuUnit;

Procedure TViewCandidateListForm.CreateParams(Var Params: TCreateParams);

Begin

Inherited;

Params.ExStyle := Params.ExStyle Or WS\_EX\_APPWINDOW;

End;

Procedure TViewCandidateListForm.FormCreate(Sender: TObject);

Begin

With CandidateListStringGrid Do

Begin

Cells[0, 0] := '№ округа';

Cells[1, 0] := 'Фамилия';

Cells[2, 0] := 'Имя';

Cells[3, 0] := 'Отчество';

Cells[4, 0] := 'Партия';

Cells[5, 0] := 'Возраст';

Cells[6, 0] := 'Профессия';

End;

End;

Function TViewCandidateListForm.FormHelp(Command: Word; Data: NativeInt; Var

CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

FormHelp := False;

End;

Procedure TViewCandidateListForm.FormShow(Sender: TObject);

Begin

CandidateListStringGrid.RowCount := CountCandidates(CandidateList) + 1;

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

End;

Procedure TViewCandidateListForm.OpenMenuItemClick(Sender: TObject);

Var

InputFile: TCandidateDataFile;

TempCandidateList: PCandidate;

IsCorrect: Boolean;

Begin

If OpenTextFileDialog.Execute Then

Begin

AssignFile(InputFile, OpenTextFileDialog.FileName);

TempCandidateList := Nil;

IsCorrect := ReadFileData(InputFile, TempCandidateList);

If IsCorrect Then

Begin

ClearCandidates(CandidateList);

RecordFileData(CandidateList,TempCandidateList,CandidateListStringGrid);

IsSaved := True;

SaveMenuItem.Enabled := True;

End

Else

Begin

ClearCandidates(TempCandidateList);

Application.MessageBox('Содержимое файла повреждено!', 'Ошибка', MB\_OK +

MB\_ICONERROR);

End;

End;

End;

Procedure TViewCandidateListForm.SaveMenuItemClick(Sender: TObject);

Var

OutputFile: TCandidateDataFile;

Begin

If SaveTextFileDialog.Execute Then

Begin

AssignFile(OutputFile, SaveTextFileDialog.FileName);

WriteFileData(OutputFile, CandidateList);

IsSaved := True;

End;

End;

Function GetCandidateData(CandidateListStringGrid: TStringGrid): TCandidateData;

Var

OldCandidateData: TCandidateData;

Begin

With OldCandidateData, CandidateListStringGrid Do

Begin

DistrictNumber := StrToInt(Cells[0, Row]);

LastName := TName(Cells[1, Row]);

FirstName := TName(Cells[2, Row]);

Patronymic := TName(Cells[3, Row]);

Party := TPartyName(Cells[4, Row]);

Age := StrToInt(Cells[5, Row]);

Profession := TProfessionName(Cells[6, Row]);

End;

GetCandidateData := OldCandidateData;

End;

Procedure CreateModifyForm(Const TitleLabelCaption, ModifyButtonCaption: String);

Begin

ModifyCandidateForm := TModifyCandidateForm.Create(ViewCandidateListForm);

ModifyCandidateForm.Icon := ViewCandidateListForm.Icon;

ModifyCandidateForm.TitleLabel.Caption := TitleLabelCaption;

ModifyCandidateForm.ModifyButton.Caption := ModifyButtonCaption;

ModifyCandidateForm.ShowModal;

End;

Procedure TViewCandidateListForm.AddButtonClick(Sender: TObject);

Begin

If CountCandidates(CandidateList) < MAX\_CANDIDATE\_COUNT Then

Begin

StateForm := Add;

CreateModifyForm('Добавление кандидата', 'Добавить');

If IsModified Then

Begin

CandidateListStringGrid.RowCount := CandidateListStringGrid.RowCount +

1;

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

IsModified := False;

IsSaved := False;

SaveMenuItem.Enabled := True;

ContentEdit.Text := '';

End;

End

Else

Application.MessageBox(PWideChar('Количество кандидатов не может превышать '

+ IntToStr(MAX\_CANDIDATE\_COUNT) + ' !'),

'Предупреждение', MB\_OK + MB\_ICONWARNING);

End;

Procedure TViewCandidateListForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_INSERT Then

AddButtonClick(AddButton)

Else If Key = VK\_ESCAPE Then

Close;

End;

Procedure TViewCandidateListForm.EditButtonClick(Sender: TObject);

Begin

If CandidateListStringGrid.Row > 0 Then

Begin

StateForm := Edit;

OldCandidateData := GetCandidateData(CandidateListStringGrid);

CreateModifyForm('Изменение кандидата', 'Изменить');

If IsModified Then

Begin

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

IsModified := False;

IsSaved := False;

SaveMenuItem.Enabled := True;

ContentEdit.Text := '';

End;

End

Else

Application.MessageBox('Не выбрано редактируемое поле!', 'Предупреждение',

MB\_OK + MB\_ICONWARNING);

End;

Procedure TViewCandidateListForm.CandidateListStringGridDblClick (Sender: TObject);

Begin

EditButtonClick(EditButton);

End;

Procedure TViewCandidateListForm.DeleteButtonClick(Sender: TObject);

Var

Confirmation, CandidateCount: Integer;

Begin

If CandidateListStringGrid.Row > 0 Then

Begin

Confirmation := Application.MessageBox('Вы действительно хотите удалить

кандидата?', 'Удаление кандидата',

MB\_YESNO + MB\_ICONQUESTION +

MB\_DEFBUTTON2);

If Confirmation = IDYES Then

Begin

DeleteCandidate(CandidateList,

GetCandidateData(CandidateListStringGrid));

CandidateListStringGrid.RowCount := CandidateListStringGrid.RowCount –

1;

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

CandidateCount := CountCandidates(CandidateList);

IsSaved := CandidateCount = 0;

SaveMenuItem.Enabled := CandidateCount > 0;

ContentEdit.Text := '';

End;

End

Else

Application.MessageBox('Не выбрано редактируемое поле!', 'Предупреждение',

MB\_OK + MB\_ICONWARNING);

End;

Procedure TViewCandidateListForm.CandidateListStringGridKeyDown(Sender: TObject; Var

Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_DELETE Then

DeleteButtonClick(DeleteButton);

End;

Procedure TViewCandidateListForm.CandidateListStringGridSelectCell(Sender: TObject;

ACol, ARow:

Integer; Var

CanSelect:

Boolean);

Begin

If ARow > 0 Then

ContentEdit.Text := CandidateListStringGrid.Cells[ACol, ARow]

Else

ContentEdit.Text := '';

End;

End.

Unit PartyUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.StdCtrls,

CandidateListUnit, PartyListUnit, Vcl.Menus, Vcl.ExtDlgs;

Type

TPartyForm = Class(TForm)

MainMenu: TMainMenu;

FileMenuItem: TMenuItem;

SaveMenuItem: TMenuItem;

SaveTextFileDialog: TSaveTextFileDialog;

ContentEdit: TEdit;

PartyStringGrid: TStringGrid;

Procedure CreateParams(Var Params: TCreateParams); Override;

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure CreateBulletinForm();

Procedure CreatePartyForm();

Procedure FormShow(Sender: TObject);

Procedure SaveMenuItemClick(Sender: TObject);

Procedure PartyStringGridSelectCell(Sender: TObject; ACol, ARow: Integer;

Var CanSelect: Boolean);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

PartyForm: TPartyForm;

Implementation

{$R \*.dfm}

Uses

MenuUnit, SearchUnit;

Procedure TPartyForm.CreateParams(Var Params: TCreateParams);

Begin

Inherited;

Params.ExStyle := Params.ExStyle Or WS\_EX\_APPWINDOW;

End;

Function TPartyForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Begin

CallHelp := False;

FormHelp := False;

End;

Procedure TPartyForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_ESCAPE Then

Close;

End;

Procedure TPartyForm.CreateBulletinForm();

Begin

PartyForm.Caption := 'Бюллетень';

With PartyStringGrid Do

Begin

ColCount := 4;

DefaultColWidth := 211;

Cells[0, 0] := 'Партия';

Cells[1, 0] := 'Заявления';

Cells[2, 0] := 'Возраст';

Cells[3, 0] := 'Профессия';

FillBulletinStringGrid(CandidateList, PartyStringGrid);

End;

End;

Procedure TPartyForm.CreatePartyForm();

Begin

PartyForm.Caption := 'Партия ' + String(PartyName);

With PartyStringGrid Do

Begin

ColCount := 6;

DefaultColWidth := 140;

Cells[0, 0] := '№ округа';

Cells[1, 0] := 'Фамилия';

Cells[2, 0] := 'Имя';

Cells[3, 0] := 'Отчество';

Cells[4, 0] := 'Возраст';

Cells[5, 0] := 'Профессия';

FillPartyStringGrid(CandidateList, PartyStringGrid, PartyName);

End;

End;

Procedure TPartyForm.FormShow(Sender: TObject);

Begin

If StateForm = Bulletin Then

CreateBulletinForm()

Else

CreatePartyForm();

End;

procedure TPartyForm.SaveMenuItemClick(Sender: TObject);

Var

OutputFile: TextFile;

Col, Row: Integer;

Begin

If SaveTextFileDialog.Execute Then

Begin

AssignFile(OutputFile, SaveTextFileDialog.FileName);

Rewrite(OutputFile);

For Row := 0 To PartyStringGrid.RowCount - 1 Do

Begin

For Col := 0 To PartyStringGrid.ColCount - 1 Do

Write(OutputFile, Format('%-40s ', [PartyStringGrid.Cells[Col,

Row]]));

WriteLn(OutputFile);

End;

CloseFile(OutputFile);

IsSaved := True;

End;

end;

Procedure TPartyForm.PartyStringGridSelectCell(Sender: TObject; ACol, ARow: Integer;

Var CanSelect: Boolean);

Begin

If ARow > 0 Then

ContentEdit.Text := PartyStringGrid.Cells[ACol, ARow]

Else

ContentEdit.Text := '';

End;

End.

Unit SearchUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,

CandidateListUnit, PartyUnit, NameComponentUnit;

Type

TSearchForm = Class(TForm)

TitleLabel: TLabel;

SearchEdit: TEdit;

SearchButton: TButton;

Procedure CreateParams(Var Params: TCreateParams); override;

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure SearchEditChange(Sender: TObject);

Procedure SearchEditContextPopup(Sender: TObject; MousePos: TPoint; Var

Handled: Boolean);

Procedure SearchEditKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Procedure SearchEditKeyPress(Sender: TObject; Var Key: Char);

Procedure SearchEditKeyUp(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Procedure SearchButtonClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

SearchForm: TSearchForm;

PartyName: TPartyName;

Implementation

{$R \*.dfm}

Uses

MenuUnit;

Procedure TSearchForm.CreateParams(Var Params: TCreateParams);

Begin

Inherited;

Params.ExStyle := Params.ExStyle Or WS\_EX\_APPWINDOW;

End;

Function TSearchForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp:

Boolean): Boolean;

Begin

CallHelp := False;

FormHelp := False;

End;

Procedure TSearchForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_ESCAPE Then

Close;

End;

Procedure TSearchForm.SearchEditChange(Sender: TObject);

Begin

PartyName := TPartyName(Trim(SearchEdit.Text));

SearchButton.Enabled := Trim(SearchEdit.Text) <> '';

End;

Procedure TSearchForm.SearchEditContextPopup(Sender: TObject; MousePos: TPoint; Var

Handled: Boolean);

Begin

NameComponentContextPopup(SearchEdit.Text, SearchEdit.SelStart,

SearchEdit.SelLength, MAX\_PARTY\_NAME\_LENGTH, Handled);

End;

Procedure TSearchForm.SearchEditKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

NameComponentKeyDown(SearchEdit.Text, SearchEdit.SelStart, SearchEdit.SelLength,

MAX\_PARTY\_NAME\_LENGTH, Key, Shift);

End;

Procedure TSearchForm.SearchEditKeyPress(Sender: TObject; Var Key: Char);

Begin

If Key = ENTER Then

SearchButtonClick(SearchButton)

Else

NameComponentKeyPress(SearchEdit.Text, SearchEdit.SelStart,

SearchEdit.SelLength, MAX\_PARTY\_NAME\_LENGTH, Key);

End;

Procedure TSearchForm.SearchEditKeyUp(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

NameComponentKeyUp();

End;

Procedure TSearchForm.SearchButtonClick(Sender: TObject);

Begin

If IsExistParty(CandidateList, PartyName) Then

Begin

PartyForm := TPartyForm.Create(Self);

PartyForm.Icon := SearchForm.Icon;

PartyForm.ShowModal;

End

Else

Application.MessageBox('Такой партии не найдено!', 'Предупреждение', MB\_OK +

MB\_ICONWARNING)

End;

End.

Unit ModifyCandidateUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Grids,

CandidateListUnit, PositiveNumberComponentUnit, NameComponentUnit;

Type

TModifyCandidateForm = Class(TForm)

TitleLabel: TLabel;

DistrictNumberLabel: TLabel;

DistrictNumberEdit: TEdit;

LastNameLabel: TLabel;

LastNameEdit: TEdit;

FirstNameLabel: TLabel;

FirstNameEdit: TEdit;

PatronymicLabel: TLabel;

PatronymicEdit: TEdit;

PartyLabel: TLabel;

PartyEdit: TEdit;

AgeLabel: TLabel;

AgeEdit: TEdit;

ProfessionLabel: TLabel;

ProfessionEdit: TEdit;

ModifyButton: TButton;

CancelButton: TButton;

Procedure CreateParams(Var Params: TCreateParams); override;

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):

Boolean;

Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure FormShow(Sender: TObject);

Procedure EditChange(Sender: TObject);

Procedure ControlKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Procedure PositiveNumberEditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure PositiveNumberEditKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Procedure PositiveNumberEditKeyPress(Sender: TObject; Var Key: Char);

Procedure PositiveNumberEditKeyUp(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Procedure NameEditContextPopup(Sender: TObject; MousePos: TPoint; Var

Handled: Boolean);

Procedure NameEditKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Procedure NameEditKeyPress(Sender: TObject; Var Key: Char);

Procedure NameEditKeyUp(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure ModifyButtonClick(Sender: TObject);

Procedure CancelButtonClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

ModifyCandidateForm: TModifyCandidateForm;

Implementation

{$R \*.dfm}

Uses

MenuUnit, ViewCandidateListUnit;

Procedure SetCandidateData(OldCandidateData: TCandidateData);

Begin

With ModifyCandidateForm, OldCandidateData Do

Begin

DistrictNumberEdit.Text := IntToStr(DistrictNumber);

LastNameEdit.Text := String(LastName);

FirstNameEdit.Text := String(FirstName);

PatronymicEdit.Text := String(Patronymic);

PartyEdit.Text := String(Party);

AgeEdit.Text := IntToStr(DistrictNumber);

ProfessionEdit.Text := String(Profession);

End;

End;

Function GetCandidateData(): TCandidateData;

Var

NewCandidateData: TCandidateData;

Begin

With NewCandidateData, ModifyCandidateForm Do

Begin

DistrictNumber := StrToInt(DistrictNumberEdit.Text);

LastName := TName(Trim(LastNameEdit.Text));

FirstName := TName(Trim(FirstNameEdit.Text));

Patronymic := TName(Trim(PatronymicEdit.Text));

Party := TPartyName(Trim(PartyEdit.Text));

Age := StrToInt(AgeEdit.Text);

Profession := TProfessionName(Trim(ProfessionEdit.Text));

End;

GetCandidateData := NewCandidateData;

End;

Procedure TModifyCandidateForm.CreateParams(Var Params: TCreateParams);

Begin

Inherited;

Params.ExStyle := Params.ExStyle Or WS\_EX\_APPWINDOW;

End;

Function TModifyCandidateForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp:

Boolean): Boolean;

Begin

CallHelp := False;

FormHelp := False;

End;

Procedure TModifyCandidateForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_ESCAPE Then

Close;

End;

Procedure TModifyCandidateForm.FormShow(Sender: TObject);

Begin

TitleLabel.Left := (ClientWidth - TitleLabel.Width) Div 2;

If StateForm = Edit Then

SetCandidateData(OldCandidateData);

End;

Function IsFullFields(): Boolean;

Begin

With ModifyCandidateForm Do

IsFullFields := (Trim(DistrictNumberEdit.Text) <> '') And

(Trim(LastNameEdit.Text) <> '') And

(Trim(FirstNameEdit.Text) <> '') And

(Trim(PartyEdit.Text) <> '') And

(Trim(AgeEdit.Text) <> '') And

(Trim(ProfessionEdit.Text) <> '');

End;

Procedure TModifyCandidateForm.EditChange(Sender: TObject);

Var

NewCandidateData: TCandidateData;

Begin

ModifyButton.Enabled := False;

If IsFullFields() Then

Begin

NewCandidateData := GetCandidateData();

ModifyButton.Enabled := Not IsExistCandidate(CandidateList,

NewCandidateData);

End;

End;

Procedure TModifyCandidateForm.ControlKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_UP Then

SelectNext(TWinControl(Sender), False, True)

Else If Key = VK\_DOWN Then

SelectNext(TWinControl(Sender), True, True)

Else If (Key = Ord(ENTER)) And (ModifyButton.Enabled) Then

ModifyButtonClick(ModifyButton);

End;

Procedure TModifyCandidateForm.PositiveNumberEditContextPopup(Sender: TObject;

MousePos: TPoint; Var

Handled: Boolean);

Var

Edit: TEdit;

Max: Integer;

Begin

Edit := TEdit(Sender);

If Edit = DistrictNumberEdit Then

Max := MAX\_DISTRICT\_NUMBER

Else

Max := MAX\_AGE;

PositiveNumberComponentContextPopup(Edit.Text, Edit.SelStart, Edit.SelLength,

Max, Handled);

End;

Procedure TModifyCandidateForm.PositiveNumberEditKeyDown(Sender: TObject; Var Key:

Word; Shift: TShiftState);

Var

Edit: TEdit;

Max: Integer;

Begin

Edit := TEdit(Sender);

If Edit = DistrictNumberEdit Then

Max := MAX\_DISTRICT\_NUMBER

Else

Max := MAX\_AGE;

PositiveNumberComponentKeyDown(Edit.Text, Edit.SelStart, Edit.SelLength, Max,

Key, Shift);

ControlKeyDown(Sender, Key, Shift);

End;

Procedure TModifyCandidateForm.PositiveNumberEditKeyPress(Sender: TObject; Var Key:

Char);

Var

Edit: TEdit;

Max: Integer;

Begin

Edit := TEdit(Sender);

If Edit = DistrictNumberEdit Then

Max := MAX\_DISTRICT\_NUMBER

Else

Max := MAX\_AGE;

PositiveNumberComponentKeyPress(Edit.Text, Edit.SelStart, Edit.SelLength, Max,

Key);

End;

Procedure TModifyCandidateForm.PositiveNumberEditKeyUp(Sender: TObject; Var Key:

Word; Shift: TShiftState);

Begin

PositiveNumberComponentKeyUp();

End;

Procedure TModifyCandidateForm.NameEditContextPopup(Sender: TObject; MousePos:

TPoint; Var Handled: Boolean);

Var

Edit: TEdit;

Max: Integer;

Begin

Edit := TEdit(Sender);

If (Edit = LastNameEdit) Or (Edit = FirstNameEdit) Or (Edit = PatronymicEdit)

Then

Max := MAX\_NAME\_LENGTH

Else If (Edit = PartyEdit) Then

Max := MAX\_PARTY\_NAME\_LENGTH

Else

Max := MAX\_PROFESSION\_NAME\_LENGTH;

NameComponentContextPopup(Edit.Text, Edit.SelStart, Edit.SelLength, Max,

Handled);

End;

Procedure TModifyCandidateForm.NameEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Var

Edit: TEdit;

Max: Integer;

Begin

Edit := TEdit(Sender);

If (Edit = LastNameEdit) Or (Edit = FirstNameEdit) Or (Edit = PatronymicEdit)

Then

Max := MAX\_NAME\_LENGTH

Else If (Edit = PartyEdit) Then

Max := MAX\_PARTY\_NAME\_LENGTH

Else

Max := MAX\_PROFESSION\_NAME\_LENGTH;

NameComponentKeyDown(Edit.Text, Edit.SelStart, Edit.SelLength, Max, Key, Shift);

ControlKeyDown(Sender, Key, Shift);

End;

Procedure TModifyCandidateForm.NameEditKeyPress(Sender: TObject; Var Key: Char);

Var

Edit: TEdit;

Max: Integer;

Begin

Edit := TEdit(Sender);

If (Edit = LastNameEdit) Or (Edit = FirstNameEdit) Or (Edit = PatronymicEdit)

Then

Max := MAX\_NAME\_LENGTH

Else If (Edit = PartyEdit) Then

Max := MAX\_PARTY\_NAME\_LENGTH

Else

Max := MAX\_PROFESSION\_NAME\_LENGTH;

NameComponentKeyPress(Edit.Text, Edit.SelStart, Edit.SelLength, Max, Key);

End;

Procedure TModifyCandidateForm.NameEditKeyUp(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

NameComponentKeyUp();

End;

Procedure TModifyCandidateForm.ModifyButtonClick(Sender: TObject);

Var

NewCandidateData: TCandidateData;

Begin

NewCandidateData := GetCandidateData();

If StateForm = Add Then

AddCandidate(CandidateList, NewCandidateData)

Else

EditCandidate(CandidateList, OldCandidateData, NewCandidateData);

IsModified := True;

Close;

End;

Procedure TModifyCandidateForm.CancelButtonClick(Sender: TObject);

Begin

Close;

End;

End.

Unit CandidateListUnit;

Interface

Uses

System.SysUtils, Vcl.Grids;

Const

MAX\_CANDIDATE\_COUNT = 1000000;

MIN\_DISTRICT\_NUMBER = 1;

MAX\_DISTRICT\_NUMBER = 100000;

MAX\_NAME\_LENGTH = 30;

MAX\_PARTY\_NAME\_LENGTH = 40;

MIN\_AGE = 1;

MAX\_AGE = 100;

MAX\_PROFESSION\_NAME\_LENGTH = 30;

Type

TDistrictNumber = Integer;

TName = String[MAX\_NAME\_LENGTH];

TPartyName = String[MAX\_PARTY\_NAME\_LENGTH];

TAge = Integer;

TProfessionName = String[MAX\_PROFESSION\_NAME\_LENGTH];

TCandidateData = Record

DistrictNumber: TDistrictNumber;

LastName: TName;

FirstName: TName;

Patronymic: TName;

Party: TPartyName;

Age: TAge;

Profession: TProfessionName;

End;

PCandidate = ^TCandidate;

TCandidate = Record

Data: TCandidateData;

Next: PCandidate;

End;

Function CountCandidates(Const CandidateList: PCandidate) : Integer;

Function IsExistCandidate(Const CandidateList: PCandidate; Const CandidateData:

TCandidateData): Boolean;

Function IsExistParty(Const CandidateList: PCandidate; Const PartyName: TPartyName):

Boolean;

Procedure AddCandidate(Var CandidateList: PCandidate; Const CandidateData:

TCandidateData);

Procedure EditCandidate(Const CandidateList: PCandidate; Const OldCandidateData,

NewCandidateData: TCandidateData);

Procedure DeleteCandidate(Var CandidateList: PCandidate; Const CandidateData:

TCandidateData);

Procedure ClearCandidates(Var CandidateList: PCandidate);

Procedure FillCandidateListStringGrid(CandidateList: PCandidate;

CandidateListStringGrid: TStringGrid);

Procedure FillPartyStringGrid(CandidateList: PCandidate; PartyStringGrid:

TStringGrid; PartyName: TPartyName);

Implementation

Function CountCandidates(Const CandidateList: PCandidate) : Integer;

Var

CurrCandidate: PCandidate;

CandidateCount: Integer;

Begin

CurrCandidate := CandidateList;

CandidateCount := 0;

While CurrCandidate <> Nil Do

Begin

Inc(CandidateCount);

CurrCandidate := CurrCandidate^.Next;

End;

CountCandidates := CandidateCount;

End;

Function IsEqualCandidates(Const CandidateData1, CandidateData2: TCandidateData):

Boolean;

Begin

IsEqualCandidates := (CandidateData1.DistrictNumber =

CandidateData2.DistrictNumber) And

(CandidateData1.LastName = CandidateData2.LastName) And

(CandidateData1.FirstName = CandidateData2.FirstName) And

(CandidateData1.Patronymic = CandidateData2.Patronymic) And

(CandidateData1.Party = CandidateData2.Party) And

(CandidateData1.Age = CandidateData2.Age) And

(CandidateData1.Profession = CandidateData2.Profession);

End;

Function SearchCandidate(Const CandidateList: PCandidate; Const CandidateData:

TCandidateData) : PCandidate;

Var

CurrCandidate: PCandidate;

Begin

CurrCandidate := CandidateList;

While Not IsEqualCandidates(CurrCandidate^.Data, CandidateData) Do

CurrCandidate := CurrCandidate^.Next;

SearchCandidate := CurrCandidate;

End;

Function IsExistCandidate(Const CandidateList: PCandidate; Const CandidateData:

TCandidateData): Boolean;

Var

CurrCandidate: PCandidate;

IsExist: Boolean;

Begin

CurrCandidate := CandidateList;

IsExist := False;

If CurrCandidate <> Nil Then

Repeat

IsExist := IsEqualCandidates(CurrCandidate^.Data, CandidateData);

CurrCandidate := CurrCandidate^.Next;

Until IsExist Or (CurrCandidate = Nil);

IsExistCandidate := IsExist;

End;

Function IsExistParty(Const CandidateList: PCandidate; Const PartyName: TPartyName):

Boolean;

Var

CurrCandidate: PCandidate;

IsExist: Boolean;

Begin

CurrCandidate := CandidateList;

IsExist := False;

If CurrCandidate <> Nil Then

Repeat

IsExist := CurrCandidate^.Data.Party = PartyName;

CurrCandidate := CurrCandidate^.Next;

Until IsExist Or (CurrCandidate = Nil);

IsExistParty := IsExist;

End;

Function CreateCandidate(Const CandidateData: TCandidateData): PCandidate;

Var

NewCandidate: PCandidate;

Begin

New(NewCandidate);

NewCandidate^.Data := CandidateData;

NewCandidate^.Next := Nil;

CreateCandidate := NewCandidate;

End;

Procedure AddCandidate(Var CandidateList: PCandidate; Const CandidateData:

TCandidateData);

Var

NewCandidate, CurrCandidate: PCandidate;

Begin

NewCandidate := CreateCandidate(CandidateData);

If CandidateList = Nil Then

CandidateList := NewCandidate

Else

Begin

CurrCandidate := CandidateList;

While CurrCandidate^.Next <> Nil Do

CurrCandidate := CurrCandidate^.Next;

CurrCandidate^.Next := NewCandidate;

End;

End;

Procedure EditCandidate(Const CandidateList: PCandidate; Const OldCandidateData,

NewCandidateData: TCandidateData);

Var

CurrCandidate: PCandidate;

Begin

CurrCandidate := SearchCandidate(CandidateList, OldCandidateData);

CurrCandidate^.Data := NewCandidateData;

End;

Procedure DeleteCandidate(Var CandidateList: PCandidate; Const CandidateData:

TCandidateData);

Var

CurrCandidate, TempCandidate: PCandidate;

Begin

CurrCandidate := CandidateList;

If IsEqualCandidates(CurrCandidate^.Data, CandidateData) Then

Begin

CandidateList := CurrCandidate^.Next;

Dispose(CurrCandidate);

End

Else

Begin

While Not IsEqualCandidates(CurrCandidate^.Next^.Data, CandidateData) Do

CurrCandidate := CurrCandidate^.Next;

TempCandidate := CurrCandidate^.Next;

CurrCandidate^.Next := CurrCandidate^.Next^.Next;

Dispose(TempCandidate);

End;

End;

Procedure ClearCandidates(Var CandidateList: PCandidate);

Var

CurrCandidate, TempCandidate: PCandidate;

Begin

CurrCandidate := CandidateList;

While CurrCandidate <> Nil Do

Begin

TempCandidate := CurrCandidate;

CurrCandidate := CurrCandidate^.Next;

Dispose(TempCandidate);

End;

CandidateList := Nil;

End;

Procedure FillCandidateListStringGrid(CandidateList: PCandidate;

CandidateListStringGrid: TStringGrid);

Var

CurrCandidate: PCandidate;

CurrRow: Integer;

Begin

CurrCandidate := CandidateList;

CurrRow := 1;

While CurrCandidate <> Nil Do

Begin

With CandidateListStringGrid, CurrCandidate^.Data Do

Begin

Cells[0, CurrRow] := IntToStr(DistrictNumber);

Cells[1, CurrRow] := String(LastName);

Cells[2, CurrRow] := String(FirstName);

Cells[3, CurrRow] := String(Patronymic);

Cells[4, CurrRow] := String(Party);

Cells[5, CurrRow] := IntToStr(Age);

Cells[6, CurrRow] := String(Profession);

End;

Inc(CurrRow);

CurrCandidate := CurrCandidate^.Next;

End;

End;

Procedure FillPartyStringGrid(CandidateList: PCandidate; PartyStringGrid:

TStringGrid; PartyName: TPartyName);

Var

CurrCandidate: PCandidate;

CurrRow: Integer;

Begin

CurrCandidate := CandidateList;

PartyStringGrid.RowCount := 1;

CurrRow := 1;

While CurrCandidate <> Nil Do

Begin

If CurrCandidate^.Data.Party = PartyName Then

Begin

With PartyStringGrid, CurrCandidate^.Data Do

Begin

RowCount := CurrRow + 1;

Cells[0, CurrRow] := IntToStr(DistrictNumber);

Cells[1, CurrRow] := String(LastName);

Cells[2, CurrRow] := String(FirstName);

Cells[3, CurrRow] := String(Patronymic);

Cells[4, CurrRow] := IntToStr(Age);

Cells[5, CurrRow] := String(Profession);

End;

Inc(CurrRow);

End;

CurrCandidate := CurrCandidate^.Next;

End;

End;

End.

Unit PartyListUnit;

Interface

Uses

System.SysUtils, Vcl.Grids,

CandidateListUnit,

ProfessionListUnit;

Type

TApplicationCount = Integer;

TAge = Integer;

TPartyData = Record

PartyName: TPartyName;

ApplicationCount: TApplicationCount;

Age: TAge;

ProfessionList: PProfession;

End;

PParty = ^TParty;

TParty = Record

Data: TPartyData;

Next: PParty;

End;

Procedure FillBulletinStringGrid(CandidateList: PCandidate; BulletinStringGrid:

TStringGrid);

Implementation

Function CreateParty(CandidateData: TCandidateData) : PParty;

Var

NewParty: PParty;

Begin

New(NewParty);

NewParty^.Data.PartyName := CandidateData.Party;

NewParty^.Data.ApplicationCount := 1;

NewParty^.Data.Age := CandidateData.Age;

NewParty^.Data.ProfessionList := Nil;

AddProfession(NewParty^.Data.ProfessionList, CandidateData.Profession);

NewParty^.Next := Nil;

CreateParty := NewParty;

End;

Procedure AddParty(Var PartyList: PParty; CandidateData: TCandidateData);

Var

NewParty, CurrParty: PParty;

Begin

NewParty := CreateParty(CandidateData);

If PartyList = Nil Then

PartyList := NewParty

Else

Begin

CurrParty := PartyList;

While (CurrParty^.Next <> Nil) And (CurrParty^.Data.PartyName <>

CandidateData.Party) Do

CurrParty := CurrParty^.Next;

If CurrParty^.Data.PartyName = CandidateData.Party Then

Begin

Inc(CurrParty^.Data.ApplicationCount);

Inc(CurrParty^.Data.Age, CandidateData.Age);

AddProfession(CurrParty^.Data.ProfessionList, CandidateData.Profession);

End

Else

CurrParty^.Next := NewParty;

End;

End;

Procedure ClearParties(Var PartyList: PParty);

Var

CurrParty, TempParty: PParty;

Begin

CurrParty := PartyList;

While CurrParty <> Nil Do

Begin

ClearProfessions(CurrParty^.Data.ProfessionList);

TempParty := CurrParty;

CurrParty := CurrParty^.Next;

Dispose(TempParty);

End;

PartyList := Nil;

End;

Function CreatePartyList(CandidateList: PCandidate) : PParty;

Var

CurrCandidate: PCandidate;

PartyList: PParty;

Begin

CurrCandidate := CandidateList;

PartyList := Nil;

While CurrCandidate <> Nil Do

Begin

AddParty(PartyList, CurrCandidate^.Data);

CurrCandidate := CurrCandidate^.Next;

End;

CreatePartyList := PartyList;

End;

Procedure FillBulletinStringGrid(CandidateList: PCandidate; BulletinStringGrid:

TStringGrid);

Var

PartyList: PParty;

CurrParty: PParty;

CurrRow: Integer;

Begin

PartyList := CreatePartyList(CandidateList);

BulletinStringGrid.RowCount := 1;

CurrParty := PartyList;

CurrRow := 1;

While CurrParty <> Nil Do

Begin

With BulletinStringGrid, CurrParty^.Data Do

Begin

RowCount := CurrRow + 1;

Cells[0, CurrRow] := String(PartyName);

Cells[1, CurrRow] := IntToStr(ApplicationCount);

Cells[2, CurrRow] := IntToStr(Age Div ApplicationCount);

Cells[3, CurrRow] := String(FindPopularProfession(ProfessionList));

CurrParty := CurrParty^.Next;

End;

Inc(CurrRow);

End;

ClearParties(PartyList);

End;

End.

Unit ProfessionListUnit;

Interface

Uses

CandidateListUnit;

Type

TProfessionData = Record

ProfessionName: TProfessionName;

Count: Integer;

End;

PProfession = ^TProfession;

TProfession = Record

Data: TProfessionData;

Next: PProfession;

End;

Procedure AddProfession(Var ProfessionList: PProfession; ProfessionName:

TProfessionName);

Procedure ClearProfessions(Var ProfessionList: PProfession);

Function FindPopularProfession(ProfessionList: PProfession) : TProfessionName;

Implementation

Function CreateProfession(ProfessionName: TProfessionName) : PProfession;

Var

NewProfession: PProfession;

Begin

New(NewProfession);

NewProfession^.Data.ProfessionName := ProfessionName;

NewProfession^.Data.Count := 1;

NewProfession^.Next := Nil;

CreateProfession := NewProfession;

End;

Procedure AddProfession(Var ProfessionList: PProfession; ProfessionName:

TProfessionName);

Var

NewProfession, CurrProfession: PProfession;

Begin

NewProfession := CreateProfession(ProfessionName);

If ProfessionList = Nil Then

ProfessionList := NewProfession

Else

Begin

CurrProfession := ProfessionList;

While (CurrProfession^.Next <> Nil) And (CurrProfession^.Data.ProfessionName

<> ProfessionName) Do

CurrProfession := CurrProfession^.Next;

If CurrProfession^.Data.ProfessionName = ProfessionName Then

Inc(CurrProfession.Data.Count)

Else

CurrProfession^.Next := NewProfession;

End;

End;

Procedure ClearProfessions(Var ProfessionList: PProfession);

Var

CurrProfession, TempProfession: PProfession;

Begin

CurrProfession := ProfessionList;

While CurrProfession <> Nil Do

Begin

TempProfession := CurrProfession;

CurrProfession := CurrProfession^.Next;

Dispose(TempProfession);

End;

ProfessionList := Nil;

End;

Function FindPopularProfession(ProfessionList: PProfession) : TProfessionName;

Var

CurrProfession: PProfession;

PopularProfessionData: TProfessionData;

Begin

CurrProfession := ProfessionList;

PopularProfessionData := CurrProfession^.Data;

While CurrProfession <> Nil Do

Begin

If PopularProfessionData.Count < CurrProfession^.Data.Count Then

PopularProfessionData := CurrProfession^.Data;

CurrProfession := CurrProfession^.Next;

End;

FindPopularProfession := PopularProfessionData.ProfessionName;

End;

End.

Unit FileUnit;

Interface

Uses

Vcl.Grids,

CandidateListUnit;

Type

TCandidateDataFile = File Of TCandidateData;

Function ReadFileData(Var InputFile: TCandidateDataFile; Var TempCandidateList:

PCandidate): Boolean;

Procedure RecordFileData(Var CandidateList: PCandidate; Const TempCandidateList:

PCandidate; CandidateListStringGrid: TStringGrid);

Procedure WriteFileData(Var OutputFile: TCandidateDataFile; Const CandidateList:

PCandidate);

Implementation

Function IsCorrectFileData(Const TempCandidateData: TCandidateData) : Boolean;

Begin

With TempCandidateData Do

IsCorrectFileData := (DistrictNumber >= MIN\_DISTRICT\_NUMBER) And

(DistrictNumber <= MAX\_DISTRICT\_NUMBER) And

(Length(LastName) >= 1) And (Length(LastName) <=

MAX\_NAME\_LENGTH) And

(Length(FirstName) >= 1) And (Length(FirstName) <=

MAX\_NAME\_LENGTH) And

(Length(Patronymic) <= MAX\_NAME\_LENGTH) And

(Length(Party) >= 1) And (Length(Party) <=

MAX\_PARTY\_NAME\_LENGTH) And

(Age >= MIN\_AGE) And (Age <= MAX\_AGE) And

(Length(Profession) >= 1) And (Length(Profession) <=

MAX\_PROFESSION\_NAME\_LENGTH);

End;

Function ReadFileData(Var InputFile: TCandidateDataFile; Var TempCandidateList:

PCandidate): Boolean;

Var

IsCorrect: Boolean;

CandidateCount: Integer;

TempCandidateData: TCandidateData;

Begin

Reset(InputFile);

IsCorrect := True;

CandidateCount := 0;

While IsCorrect And (CandidateCount < MAX\_CANDIDATE\_COUNT) And Not

EOF(InputFile) Do

Begin

Read(InputFile, TempCandidateData);

IsCorrect := IsCorrectFileData(TempCandidateData) And Not

IsExistCandidate(TempCandidateList, TempCandidateData);

If IsCorrect Then

Begin

AddCandidate(TempCandidateList, TempCandidateData);

Inc(CandidateCount);

End;

End;

If IsCorrect Then

IsCorrect := CandidateCount <= MAX\_CANDIDATE\_COUNT;

CloseFile(InputFile);

ReadFileData := IsCorrect;

End;

Procedure RecordFileData(Var CandidateList: PCandidate; Const TempCandidateList:

PCandidate; CandidateListStringGrid: TStringGrid);

Begin

CandidateList := TempCandidateList;

CandidateListStringGrid.RowCount := CountCandidates(CandidateList) + 1;

FillCandidateListStringGrid(CandidateList, CandidateListStringGrid);

End;

Procedure WriteFileData(Var OutputFile: TCandidateDataFile; Const CandidateList:

PCandidate);

Var

CurrCandidate: PCandidate;

Begin

Rewrite(OutputFile);

CurrCandidate := CandidateList;

While CurrCandidate <> Nil Do

Begin

Write(OutputFile, CurrCandidate^.Data);

CurrCandidate := CurrCandidate^.Next;

End;

CloseFile(OutputFile);

End;

End.

Unit PositiveNumberComponentUnit;

Interface

Uses

Winapi.Windows, System.SysUtils, System.Classes, Clipbrd;

Const

ENTER = #13;

BACKSPACE = #8;

NONE = #0;

DIGITS = ['0' .. '9'];

DIGITS\_WITHOUT\_ZERO = ['1' .. '9'];

ALPHABET = ['A' .. 'Z', 'a' .. 'z'];

Var

CtrlPressed: Boolean = False;

Procedure PositiveNumberComponentContextPopup(Const Text: String; Const SelStart,

SelLength, MAX: Integer; Var Handled:

Boolean);

Procedure PositiveNumberComponentExit(Text: String; Const MIN: Integer);

Procedure PositiveNumberComponentKeyDown(Const Text: String; Const SelStart,

SelLength, MAX: Integer; Var Key: Word;

Shift: TShiftState);

Procedure PositiveNumberComponentKeyPress(Const Text: String; Const SelStart,

SelLength, MAX: Integer; Var Key: Char);

Procedure PositiveNumberComponentKeyUp();

Implementation

Function IsValidRange(Const Num, MAX: Integer): Boolean;

Begin

IsValidRange := Num <= MAX;

End;

Function IsPossiblePaste(Const Text: String; Const SelStart, SelLength, MAX:

Integer): Boolean;

Var

Num: Integer;

Begin

IsPossiblePaste := Clipboard.HasFormat(CF\_TEXT) And

(Length(Clipboard.AsText) <> 0) And

TryStrToInt(Copy(Text, 1, SelStart) + Clipboard.AsText +

Copy(Text, SelStart + SelLength + 1), Num) And

((SelStart = 0) And (Clipboard.AsText[1] <> '0') Or (SelStart

> 0)) And

IsValidRange(StrToInt(Copy(Text, 1, SelStart) +

Clipboard.AsText + Copy(Text, SelStart

+ SelLength + 1)), MAX);

End;

Function IsValidChar(Const SelStart: Integer; Const Key: Char): Boolean;

Begin

IsValidChar := (SelStart = 0) And CharInSet(Key, DIGITS\_WITHOUT\_ZERO) Or

(SelStart > 0) And CharInSet(Key, DIGITS);

End;

Procedure PositiveNumberComponentContextPopup(Const Text: String; Const SelStart,

SelLength, MAX: Integer; Var

Handled: Boolean);

Begin

If Not IsPossiblePaste(Text, SelStart, SelLength, MAX) Or

(SelLength = 0) And (SelStart = 1) And (Length(Text) > 1) And (Text[2] = '0')

Or (SelLength > 0) And (SelStart = 0) And (SelLength <> Length(Text)) And

(Text[SelLength + 1] = '0') Then

Handled := True;

End;

Procedure PositiveNumberComponentExit(Text: String; Const MIN: Integer);

Begin

If (Text <> '') And (StrToInt(Text) < MIN) Then

Text := '';

End;

Procedure PositiveNumberComponentKeyDown(Const Text: String; Const SelStart,

SelLength, MAX: Integer; Var Key: Word;

Shift: TShiftState);

Begin

If (Shift = [ssCtrl]) And (UpCase(Chr(Key)) = 'X') Then

Begin

If (SelLength = 0) And (SelStart = 1) And (Length(Text) > 1) And (Text[2] =

'0') Or

(SelLength > 0) And (SelStart = 0) And (SelLength <> Length(Text)) And

(Text[SelLength + 1] = '0') Then

Key := Ord(NONE);

End

Else If Key = VK\_DELETE Then

Begin

If (SelLength = 0) And (SelStart = 0) And (Length(Text) > 1) And (Text[2] =

'0') Or

(SelLength > 0) And (SelStart = 0) And (SelLength <> Length(Text)) And

(Text[SelLength + 1] = '0') Then

Key := Ord(NONE);

End

Else If (Shift = [ssCtrl]) And (UpCase(Chr(Key)) = 'V') Or

(Shift = [ssShift]) And (Key = VK\_INSERT) Then

Begin

If Not IsPossiblePaste(Text, SelStart, SelLength, MAX) Then

Key := Ord(NONE);

End;

If (Shift = [ssCtrl]) And CharInSet(Chr(Key), ALPHABET) Then

CtrlPressed := True;

End;

Procedure PositiveNumberComponentKeyPress(Const Text: String; Const SelStart,

SelLength, MAX: Integer; Var Key: Char);

Begin

If Key = BACKSPACE Then

Begin

If (SelLength = 0) And (SelStart = 1) And (Length(Text) > 1) And (Text[2] =

'0') Or

(SelLength > 0) And (SelStart = 0) And (SelLength <> Length(Text)) And

(Text[SelLength + 1] = '0') Then

Key := NONE;

End

Else If Not CtrlPressed Then

Begin

If Not (IsValidChar(SelStart, Key) And IsValidRange(StrToInt(Copy(Text, 1,

SelStart) + Key + Copy(Text, SelStart + SelLength + 1)), MAX)) Then

Key := NONE;

End;

End;

Procedure PositiveNumberComponentKeyUp();

Begin

CtrlPressed := False;

End;

End.

Unit NameComponentUnit;

Interface

Uses

Winapi.Windows, System.SysUtils, System.Classes, Clipbrd;

Const

ENTER = #13;

BACKSPACE = #8;

NONE = #0;

ALPHABET = ['A' .. 'Z', 'a' .. 'z'];

VALID\_CHARS = ' –

абвгдеежзийклмнопрстуфхцчшщъыьэюяabcdefghijklmnopqrstuvwxyzАБВГД

ЕЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯABCDEFGHIJKLMNOPQRSTUVWXYZ';

Var

CtrlPressed: Boolean = False;

Procedure NameComponentContextPopup(Const Text: String; Const SelStart, SelLength,

MAX: Integer; Var Handled: Boolean);

Procedure NameComponentExit(Text: String; Const MIN: Integer);

Procedure NameComponentKeyDown(Const Text: String; Const SelStart, SelLength, MAX:

Integer; Var Key: Word; Shift: TShiftState);

Procedure NameComponentKeyPress(Const Text: String; Const SelStart, SelLength, MAX:

Integer; Var Key: Char);

Procedure NameComponentKeyUp();

Implementation

Function IsValidRange(Const Num, MAX: Integer): Boolean;

Begin

IsValidRange := Num <= MAX;

End;

Function IsValidChar(Const Key: Char): Boolean;

Var

C: Char;

Begin

For C In VALID\_CHARS Do

If C = Key Then

Exit(True);

IsValidChar := False;

End;

Function IsValidChars(Const Text: String) : Boolean;

Var

I: Integer;

Begin

For I := 1 To Length(Text) Do

If Not IsValidChar(Text[I]) Then

Exit(False);

IsValidChars := True;

End;

Function IsPossiblePaste(Const Text: String; Const SelStart, SelLength, MAX:

Integer): Boolean;

Begin

IsPossiblePaste := Clipboard.HasFormat(CF\_TEXT) And

(Length(Clipboard.AsText) <> 0) And

IsValidChars(Clipboard.AsText) And

IsValidRange(Length(Copy(Text, 1, SelStart) +

Clipboard.AsText + Copy(Text, SelStart +

SelLength + 1)), MAX);

End;

Procedure NameComponentContextPopup(Const Text: String; Const SelStart, SelLength,

MAX: Integer; Var Handled: Boolean);

Begin

If Not IsPossiblePaste(Text, SelStart, SelLength, MAX) Then

Handled := True;

End;

Procedure NameComponentExit(Text: String; Const MIN: Integer);

Begin

If Length(Text) < MIN Then

Text := '';

End;

Procedure NameComponentKeyDown(Const Text: String; Const SelStart, SelLength, MAX:

Integer; Var Key: Word; Shift: TShiftState);

Begin

If (Shift = [ssCtrl]) And (UpCase(Chr(Key)) = 'V') Or

(Shift = [ssShift]) And (Key = VK\_INSERT) Then

Begin

If Not IsPossiblePaste(Text, SelStart, SelLength, MAX) Then

Key := Ord(NONE);

End;

If (Shift = [ssCtrl]) And CharInSet(Chr(Key), ALPHABET) Then

CtrlPressed := True;

End;

Procedure NameComponentKeyPress(Const Text: String; Const SelStart, SelLength, MAX:

Integer; Var Key: Char);

Begin

If Not CtrlPressed And (Key <> BACKSPACE) Then

Begin

If Not (IsValidChar(Key) And IsValidRange(Length(Copy(Text, 1, SelStart) +

Key + Copy(Text, SelStart + SelLength + 1)), MAX)) Then

Key := NONE;

End;

End;

Procedure NameComponentKeyUp();

Begin

CtrlPressed := False;

End;

End.