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 15

Студент Городко К. Е.

Руководитель Данилова Г.В.

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Каждая программа работает с данными, находящимися в оперативной памяти компьютера. Программисту необходимо уметь управлять памятью для наибольшей эффективности программы. Существует два основных способа распределения памяти: статический и динамический. Оба из них имеют свои преимущества и недостатки.

Статическая память выделяется во время компиляции и предоставляет быстрый доступ к данным программы. Однако это требует от программиста знания заранее, какое количество памяти потребуется. Также при использовании больших структур данных статическая память может быть неэффективно расходована.

Динамическое распределение памяти позволяет программисту управлять использующейся памятью напрямую путем ее выделения и освобождения. Это удобно в случаях, когда заранее невозможно определить объем данных, с которыми будет работать программа. Недостатками данного способа является медленный доступ к данным, а также необходимость освобождения неиспользуемых данных, что в обратном случае может привести к утечке памяти.

Для доступа к данным, находящимся в динамической памяти, используются переменные типа указатель. В таких переменных находятся не сами данные, а их адрес. Существует 2 типа указателей – типизированные (связанные с заданным в программе типом) и нетипизированные. Память для самих указателей выделяется статически.

Динамические структуры данных – структуры, элементы которых создаются и уничтожаются во время программы. Такие структуры могут не только изменять количество составляющих их элементов, но и характер связей между ними. Одной из основных динамических структур является список – набор упорядоченных элементов, связанных между собой указателями.

Целью данной учебной практики является разработка программного средства «Биржа труда» с использованием динамических списков.

1 ПОСТАНОВКА ЗАДАЧИ

В рамках данной учебной практики планируется разработать приложение «Биржа труда», которое позволит хранить данные о вакансиях и кандидатах.

Каждая запись списка с вакансиями содержит:

– название фирмы;
– наименование специальности;
– должность;
– оклад;
– количество дней отпуска;
– наличие высшего образования у нанимаемого (да/нет);
– возрастной диапазон нанимаемого (min/max).

Каждая запись списка кандидатов содержит:

– ФИО кандидата;
– дату рождения;
– специальность;
– наличие высшего образования (да/нет);
– желаемую должность;
– минимальный оклад.

Функциональные требования:

– подбор возможных кандидатов по каждой вакансии;
– формирование списка дефицитных специалистов (количество кандидатов на должность ниже 10% от количества вакансий);
– подсистема добавления, удаления и корректирования записей списков;
– подсистема просмотра списков целиком.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 11.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Структура программы

При разработке приложения будет использовано семь модулей:

– MainUnit – модуль, обеспечивающий отображение главного меню приложения;

– VacancyListUnit – модуль, отвечающий за отображение окна списка с вакансиями;

– VacancyUnit – модуль, обеспечивающий редактирование отдельной вакансии списка;

– CandidateListUnit – модуль, обеспечивающий отображение окна списка с кандидатами;

– CandidateUnit – модуль, обеспечивающий редактирование отдельного кандидата списка;

– CandidateSelectUnit – модуль, обеспечивающий подбор кандидатов по выбранной вакансии;

– DeficitUnit – модуль, формирующий и отображающий список дефицитных кандидатов.

2.2 Проектирование интерфейса программного средства

Интерфейс – это визуальное представление программного средства, с которым взаимодействует пользователь.

Интерфейс должен соответствовать следующим требованиям:

– простая и интуитивная для пользователя навигация;

– удобное расположение кнопок и элементов управления.

2.2.1 Главное окно

Главное окно приложения состоит из трех кнопок, с помощью которых пользователь может перейти к другому окну для работы с одним из списков, либо выйти из программы. Макет данного окна представлен на рисунке 2.1.

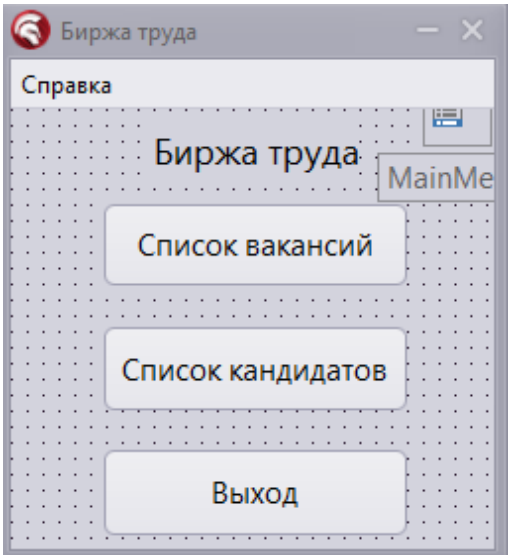

Рисунок 2.1 – Макет главного окна приложения

2.2.2 Окно просмотра списка с вакансиями

Окно просмотра списка вакансий состоит из кнопок для работы с записями списка. В верхней части окна находятся 3 кнопки:

– добавить вакансию;
– удалить вакансию;
– подобрать кандидатов.

Отображение списка осуществляется с помощью компонента TListView. В таблице находятся следующие поля:

– название фирмы;
– специальность;
– должность;
– оклад;
– количество дней отпуска;
– наличие высшего образования;
– возрастной диапазон.

Макет окна со списком вакансий представлен на рисунке 2.2.

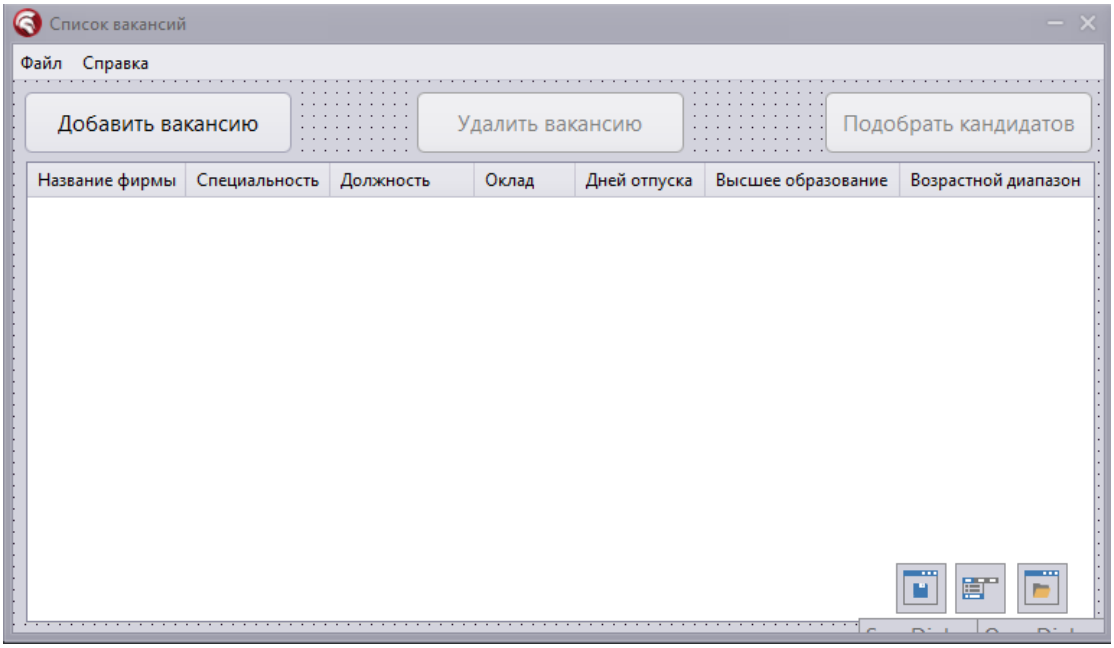

Рисунок 2.2 – Макет окна просмотра списка вакансий

2.2.3 Окно просмотра списка с кандидатами

Окно со списком кандидатов состоит из трех кнопок и таблицы, в которой будут находиться записи списка. В верхней части окна находятся следующие кнопки:

– добавить кандидата;
– удалить кандидата;
– вывести список дефицитных кандидатов.

В таблице присутствует 8 полей:

– фамилия;
– имя;
– отчество;
– дата рождения;
– специальность;
– высшее образование;
– должность;
– оклад.

Макет данного окна представлен на рисунке 2.3.

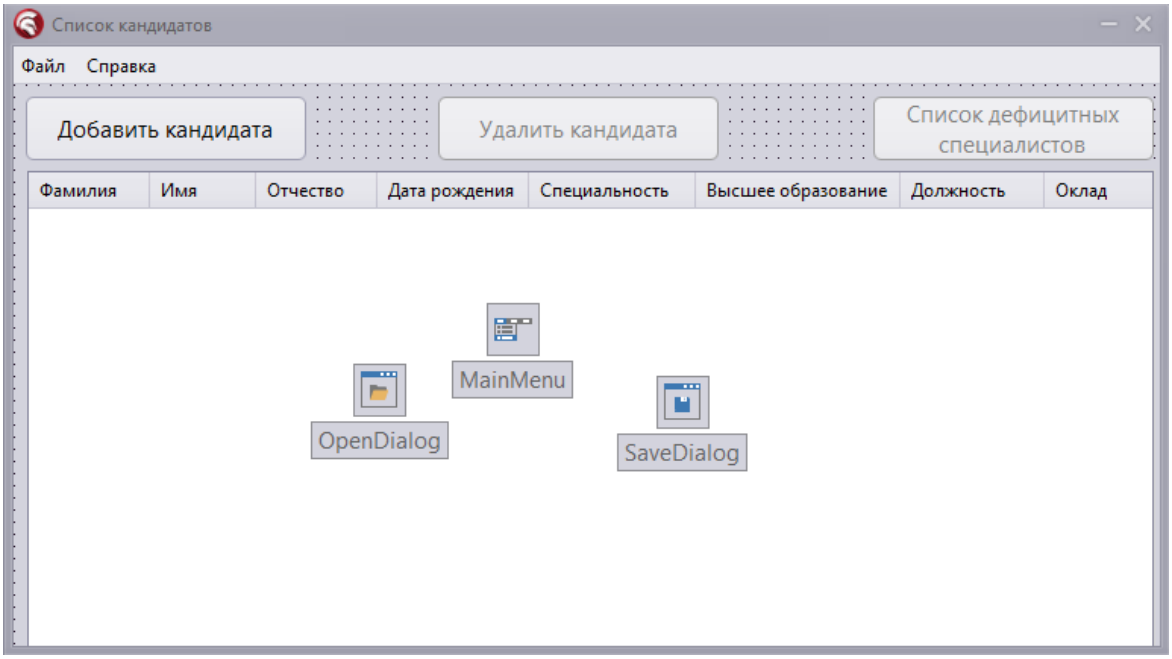

Рисунок 2.3 – Макет окна списка кандидатов

2.2.4 Окно изменения информации о вакансии

Окно изменения информации о вакансии состоит из нескольких полей для ввода данных и двух кнопок: для сохранения изменений и отмены ввода. Макет окна представлен на рисунке 2.4.

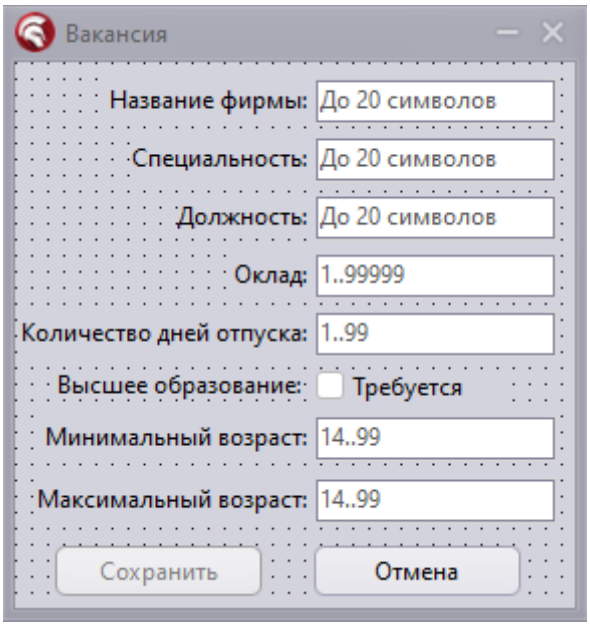

Рисунок 2.4 – Макет окна изменения информации о вакансии

2.2.5 Окно изменения информации о кандидате

Окно изменения информации о кандидате состоит из нескольких полей для ввода данных и двух кнопок. Макет окна представлен на рисунке 2.5.

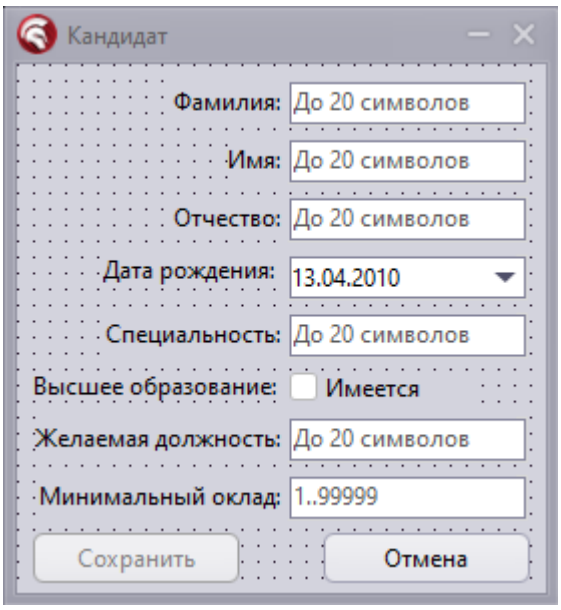

Рисунок 2.5 – Макет окна изменения информации о кандидате

2.2.6 Окно подбора кандидатов по вакансии

Окно подбора кандидатов по вакансии состоит из двух таблиц: верхняя будет использована для отображения выбранной вакансии, нижняя – для подобранных кандидатов. Макет окна подбора кандидатов изображен на рисунке 2.6.

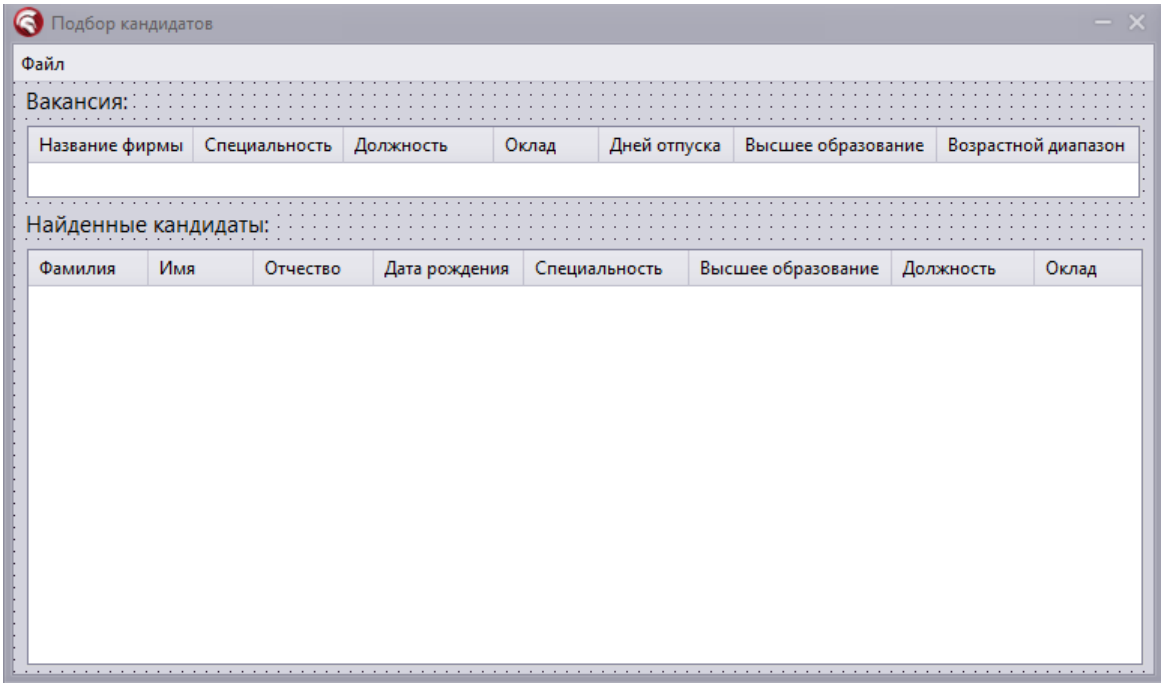

Рисунок 2.6 – Макет окна подбора кандидатов

2.2.7 Окно просмотра списка с дефицитными специалистами

Окно со списком дефицитных кандидатов состоит из таблицы с кандидатами. Макет данного окна представлен на рисунке 2.7.

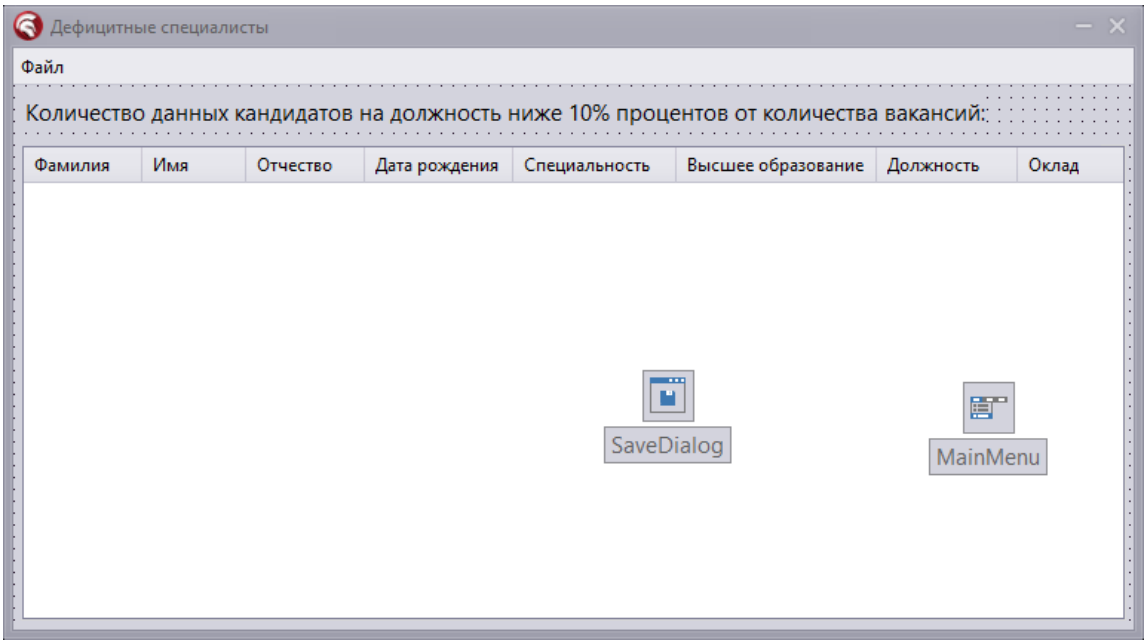

Рисунок 2.7 – Макет окна со списком дефицитных специалистов

2.3 Проектирование функционала программного средства

Для решения поставленной задачи требуется разработать алгоритмы, выполняющие основные функции программы:

– подбор возможных кандидатов по вакансии;
– формирование списка дефицитных специалистов;
– изменение записи списка;
– сохранение списка в файл.

2.3.1 Подбор возможных кандидатов по вакансии

Для подбора кандидатов по выбранной вакансии необходимо пройти по списку кандидатов, сравнивая каждую запись списка с информацией по вакансии. Если кандидат подходит по критериям вакансии, он добавляется в список найденных кандидатов и выводится в таблицу. Блок-схема данной процедуры представлена на рисунке 2.8.

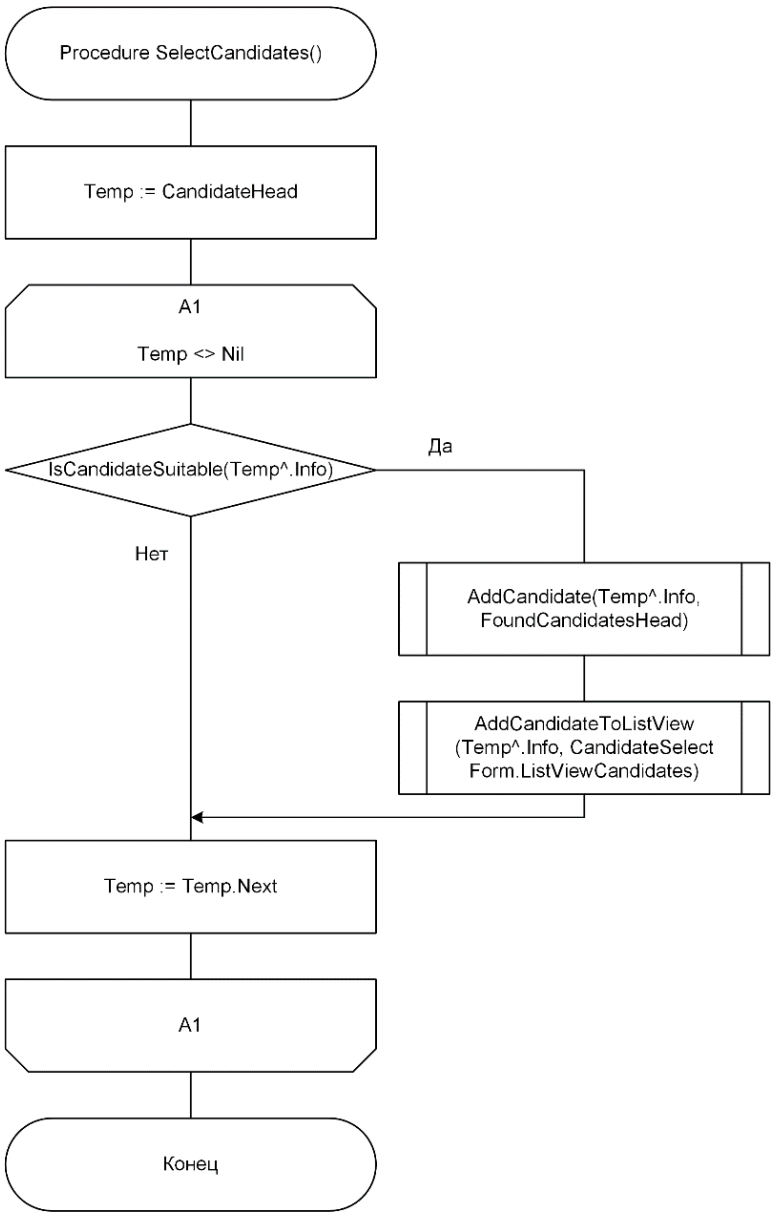

Рисунок 2.8 – Блок-схема процедуры SelectCandidates

2.3.2 Формирование списка дефицитных специалистов

Для поиска дефицитных кандидатов будет создан массив записей, каждая из которых состоит из наименования должности, а также числа вакансий и кандидатов по этой должности. В процедуре формирования списка дефицитных специалистов идет проход по всем элементам массива со сравнением количества вакансий по должности с количеством кандидатов. Если число кандидатов меньше 10% от числа вакансий, все кандидаты с данной должностью добавляются в список. Блок-схема процедуры формирования списка дефицитных специалистов представлена на рисунке 2.9.

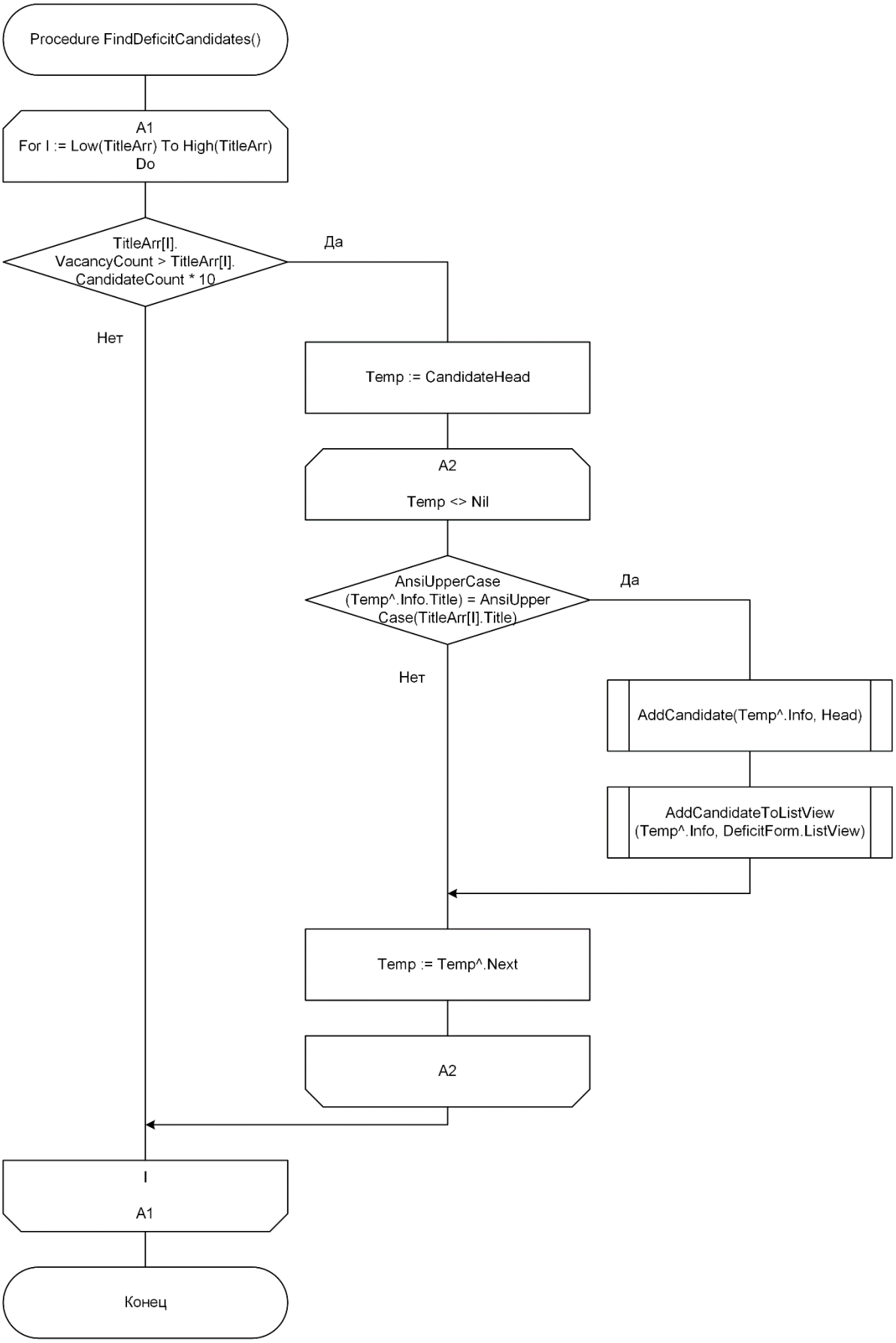

Рисунок 2.9 – Блок-схема процедуры FindDeficitCandidates

2.3.3 Изменение записи списка

Для изменения записи в списке требуется найти изменяемую запись путем прохода по списку и сравнения информации текущей записи с искомой. После нахождения записи, информация о ней корректируется и происходит вызов процедуры, обновляющей информацию о соответствующей записи в таблице. Блок-схема процедуры изменения списка вакансий представлена на рисунке 2.10. Аналогичным образом будет происходить изменение списка кандидатов.

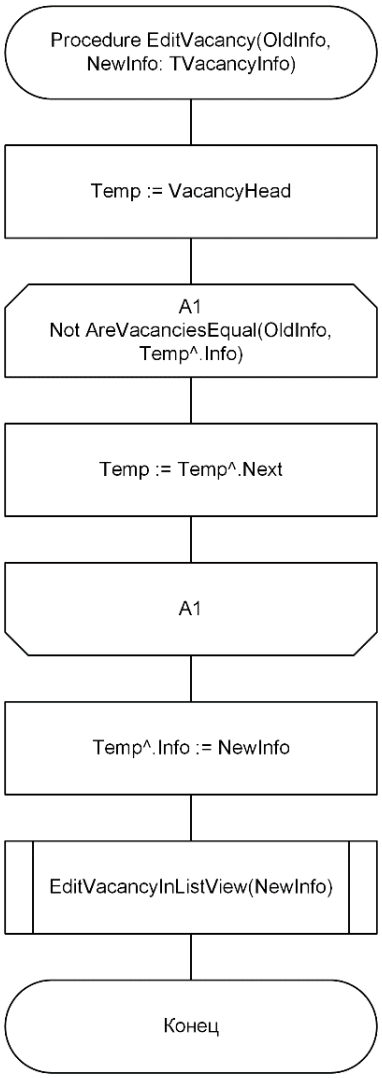

Рисунок 2.10 – Блок-схема процедуры EditVacancy

2.3.4 Сохранение списка в файл

Процедура сохранения списка в файл открывает файл по указанному пути и проходит по списку, записывая информацию по каждому узлу в файл. Блок-схема процедуры сохранения списка вакансий в файл изображена на рисунке 2.11.

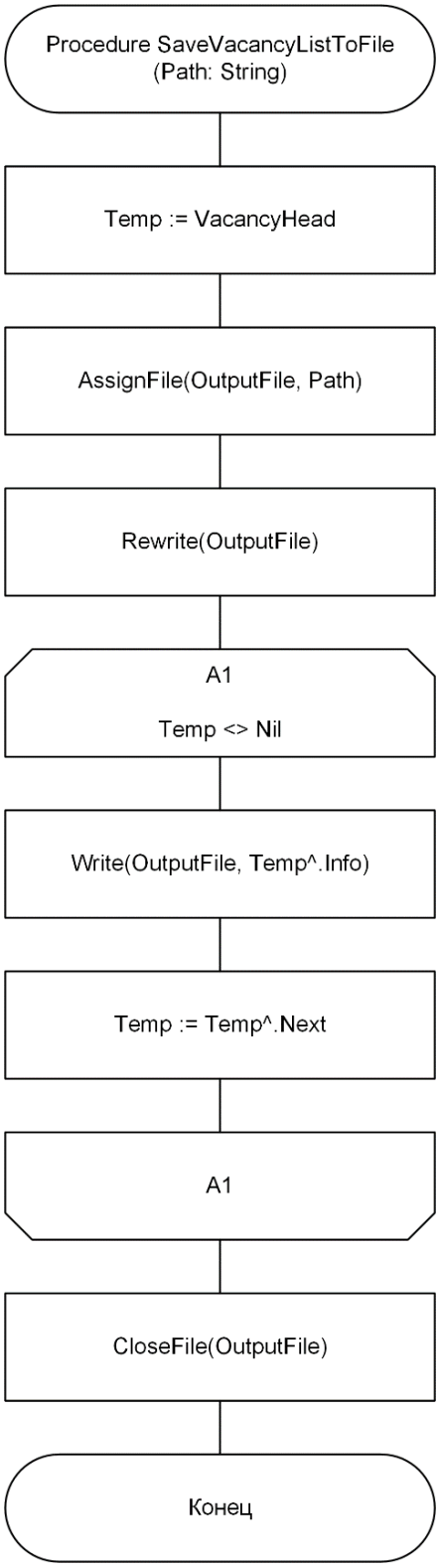

Рисунок 2.11 – Блок-схема процедуры SaveVacancyListToFile

Если файла по переданному в процедуру пути не существует, создается новый пустой файл и данные записываются в него.

3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Взаимодействие со списками

Для работы над списками была выбрана структура данных «однонаправленный список».

3.1.1 Добавление записи в список

Для добавления записи в список требуется выделить память для нового элемента и переместиться в конец списка, чтобы связать текущий последний элемент с новым. Код процедуры для добавления вакансии представлен ниже.

```
Procedure AddVacancy(VacancyInfo: TVacancyInfo; Var Head: PVacancy);
Var
    NewVacancy, Temp: PVacancy;
Begin
    // Выделение памяти для нового узла списка
    NewVacancy := New(PVacancy);
    // Запись данных в новый узел
    NewVacancy^.Info := VacancyInfo;
    NewVacancy^.Next := Nil;
    // Если список пуст
    If Head = Nil Then
        // Присваивание голове списка значения указателя на новый элемент
        Head := NewVacancy
    Else
    Begin
        // Присваивание временной переменной значения головы для дальнейшего
        // перемещения по списку
        Temp := Head;
        // Перемещение в конец списка
        While Temp^.Next <> Nil Do
            Temp := Temp^.Next;
        // Добавление нового узла в конец
        Temp^.Next := NewVacancy;
    End;
End;
```

3.1.2 Удаление списка

Для удаления всего списка необходимо пройти по списку и очистить память для каждого элемента. Код процедуры удаления списка вакансий представлен ниже.

```
Procedure DeleteVacancyList(Var Head: PVacancy);
Var
    Curr: PVacancy;
Begin
    Curr := Head;
    // Пока текущий указатель не равен нулевому
    While Curr <> Nil Do
    Begin
        // Смещение головы списка на следующий узел
```

```
        Head := Head^.Next;
        // Освобождение памяти для текущего узла
        Dispose(Curr);
        // Присваивание текущему указателю значение новой головы
        Curr := Head;
    End;
End;
```

3.2 Взаимодействие с таблицей

Для визуального представления списков был выбран компонент TListView. Взаимодействие с таблицей происходит после каждого изменения списка.

3.2.1 Добавление записи в таблицу

Для добавления записи в таблицу создается новый элемент типа TListItem, и все столбцы новой строки заполняются соответствующими данными. Код процедуры добавления вакансии в таблицу представлен ниже.

```
Procedure AddVacancyToListView(VacancyInfo: TVacancyInfo; ListView: TListView);
Var
    NewItem: TListItem;
Begin
    // Добавление нового элемента в таблицу
    NewItem := ListView.Items.Add;
    // Заполнение первого столбца
    NewItem.Caption := String(VacancyInfo.FirmName);
    // Заполнение всех остальных столбцов
    With NewItem.SubItems, VacancyInfo Do
    Begin
        Add(String(Speciality));
        Add(String(Title));
        Add(IntToStr(Salary));
        Add(IntToStr(VacationDays));
        Add(HIGHEDUCATIONREQUIRED[IsHighEducationRequired]);
        Add(IntToStr(MinAge) + '-' + IntToStr(MaxAge));
    End;
    // Изменение параметра сохранения списка
    IsVacancyListSaved := False;
End;
```

3.2.2 Очистка таблицы

Для очистки таблицы используется процедура ClearListView. Код процедуры представлен далее.

```
Procedure ClearListView(ListView: TListView);
Begin
    // Пока количество элементов таблицы не равно 0
    While ListView.Items.Count <> 0 Do
        // Удаление первой строки
        ListView.Items[0].Delete;
End;
```

3.3 Работа с файлами

В программе присутствуют функции для пользовательской работы с файлами, а именно чтение и запись данных. При работе используются типизированные файлы.

3.3.1 Сохранение списка в файл

Для сохранения списка вакансий в файл используется процедура SaveVacancyListToFile, в которую передается путь файла в виде параметра. Код процедуры представлен ниже.

```
Procedure SaveVacancyListToFile(Path: String);
Var
    OutputFile: File Of TVacancyInfo;
    Temp: PVacancy;
Begin
    Try
        // Присвоение временной переменной значение головы списка для дальнейшего
        // перемещения по списку
        Temp := VacancyHead;
        AssignFile(OutputFile, Path);
        // Создание и открытие файла
        Rewrite(OutputFile);
        // Пока не достигнут конец списка
        While Temp <> Nil Do
        Begin
            // Запись данных узла в файл
            Write(OutputFile, Temp^.Info);
            Temp := Temp^.Next;
        End;
    Finally
        CloseFile(OutputFile);
    End;
End;
```

3.3.2 Чтение списка из файла

Код метода чтения списка вакансий из файла представлен ниже.

```
Procedure TVacancyListForm.ReadVacancyListFromFile();
Var
    InputFile: File Of TVacancyInfo;
    VacancyInfo: TVacancyInfo;
    Head: PVacancy;
    IsCorrect: Boolean;
    Count: Integer;
Begin
    // Проверка на открытие диалогового окна и формат выбранного файла
    IsCorrect := OpenDialog.Execute And IsFileExtCorrect(OpenDialog.FileName,
                                                         VACANCYFILEEXT);
    If IsCorrect Then
    Begin
        Try
            // Инициализация временного списка
            Head := Nil;
            // Инициализация счетчика вакансий в списке
```

```
            Count := 0;
            AssignFile(InputFile, OpenDialog.FileName);
            Try
                // Открытие файла
                Reset(InputFile);
                // Проверка количества записей в файле
                If FileSize(InputFile) > MAXRECORDAMOUNT Then
                    IsCorrect := False;
                // Пока не достигнут конец файла и данные корректны
                While Not Eof(InputFile) And IsCorrect Do
                Begin
                    // Чтение записи
                    Read(InputFile, VacancyInfo);
                    // Проверка записи
                    IsCorrect := IsVacancyCorrect(VacancyInfo);
                    If IsCorrect then
                        // Добавление узла в список
                        AddVacancy(VacancyInfo, Head);
                    Inc(Count);
                End;
            Except
                IsCorrect := False;
            End;
        Finally
            CloseFile(InputFile);
        End;
        // Если чтение прошло успешно
        If IsCorrect Then
        Begin
            // Удаление ранее созданного списка
            DeleteVacancyList(VacancyHead);
            // Очистка таблицы
            ClearListView(ListView);
            // Присваивание голове списка значение временной головы
            VacancyHead := Head;
            While Head <> Nil Do
            Begin
                // Добавление в таблицу
                AddVacancyToListView(Head^.Info, ListView);
                Head := Head^.Next;
            End;
            VacancyAmount := Count;
            IsVacancyListSaved := True;
        End
        // Если при чтении возникли ошибки
        Else
        Begin
            // Удаление временного списка
            DeleteVacancyList(Head);
            // Вывод сообщения об ошибке
            Application.MessageBox('Произошла ошибка при открытии файла! Проверьте
                                    корректность данных!', 'Ошибка', MB_ICONERROR);
        End;
    End;
End;
```

Для прочтения списка из файла требуется открыть файл, прочитать из него данные, а также проверить их. В случае успешной проверки данные добавляются в таблицу, в случае неуспешной – память, занятая данными, освобождается, и пользователю выводится сообщение об ошибке.

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Основной проблемой было некорректное удаление. После удаления записи из списка и дальнейшей попытке работы с записями возникала ошибка неверного доступа к памяти. Это было связано с некорректным изменением структуры списка во время удаления, что, при перемещении по списку, приводило к обращению к удаленному элементу. Проблема представлена на рисунке 4.1.

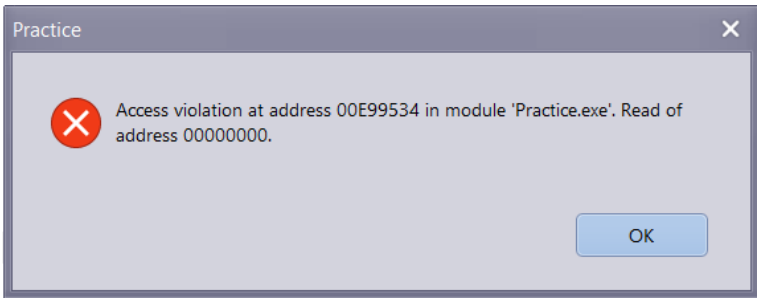

Рисунок 4.1 – Ошибка при попытке изменения списка

Проблема была решена изменением процедуры удаления из списка, которая имеет следующий вид:

```
Procedure DeleteVacancy(VacancyInfo: TVacancyInfo);
Var
    Temp, Curr: PVacancy;
Begin
    Temp := VacancyHead;
    // Если требуется удалить первый элемент списка
    If AreVacanciesEqual(VacancyInfo, Temp^.Info) Then
        // Значение головы перемещается на указатель следующего элемента
        VacancyHead := Temp^.Next
    Else
    Begin
        // Поиск элемента, идущего перед удаляемым
        While Not AreVacanciesEqual(VacancyInfo, Temp^.Next^.Info) Do
            Temp := Temp^.Next;
        Curr := Temp;
        // Присваивание временной переменной адреса удаляемой записи
        Temp := Temp^.Next;
        // Изменение структуры списка для удаления из нее требуемого узла
        Curr^.Next := Curr^.Next^.Next;
    End;
    // Освобождение памяти, занятой удаляемой записью
    Dispose(Temp);
End;
```

Большинство проблем было связано с недочетами на стадии разработки программного средства. На стадии тестирования все проблемы были исправлены.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

5.1 Интерфейс программного средства

5.1.1 Главное окно

Главное окно приложения состоит из трех кнопок:

– первой расположена кнопка «Список вакансий», которая открывает окно просмотра списка с вакансиями;

– второй является кнопка «Список кандидатов», нажатие на которую открывает окно просмотра списка с кандидатами;

– третьей расположена кнопка «Выход», закрывающая приложение.

Также в верхней части окна доступна справка с инструкцией и информацией о программе. Внешний вид главного окна представлен на рисунке 5.1.

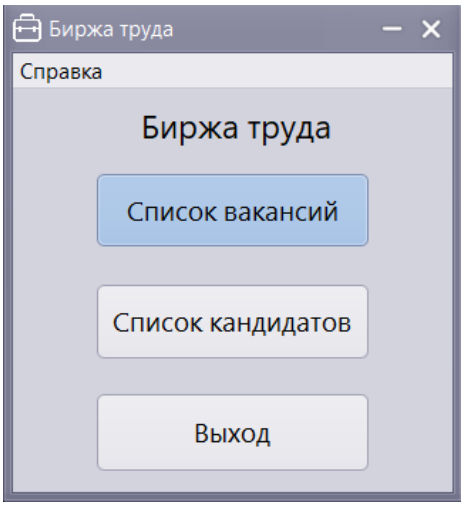

Рисунок 5.1 – Главное окно

5.1.2 Окно просмотра списка с вакансиями

В верхней части окна со списком вакансий находится меню, через которое пользователь может открыть или сохранить файл, а также открыть справку. Кнопка сохранения неактивна, если список пуст. Ниже располагаются таблица для списка и кнопки для работы над вакансиями:

– «Добавить вакансию» открывает окно изменения информации о вакансии;

– «Удалить вакансию» позволяет удалить выбранную в списке вакансию;

– «Подобрать кандидатов» открывает окно подбора кандидатов по вакансии.

Кнопки для удаления и подбора кандидатов становятся активными только после выбора вакансии в списке путем нажатия на нее. Внешний вид окна для просмотра списка вакансий представлен на рисунке 5.2.

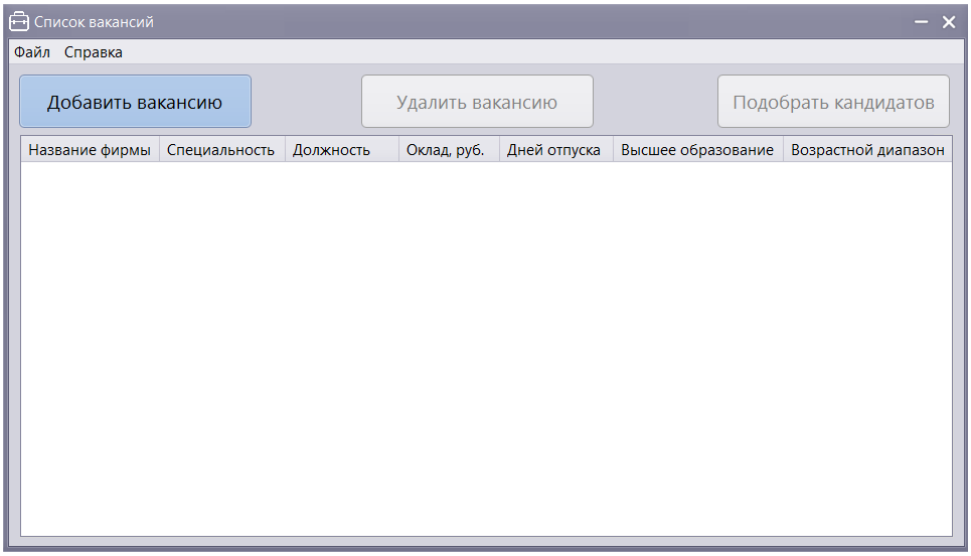

Рисунок 5.2 – Окно просмотра списка с вакансиями

5.1.3 Окно просмотра списка с кандидатами

В верхней части окна находится меню, через которое можно открыть и сохранить файл, а также открыть справку. Ниже расположены таблица, а также три кнопки:

– «Добавить кандидата», при нажатии на которую открывается окно изменения информации о кандидате;

– «Удалить кандидата», для удаления выбранного в списке кандидата;

– «Список дефицитных специалистов», открывающая окно просмотра списка дефицитных специалистов. Данная кнопка неактивна, если список пуст.

Оформление окна просмотра списка кандидатов представлено на рисунке 5.3.

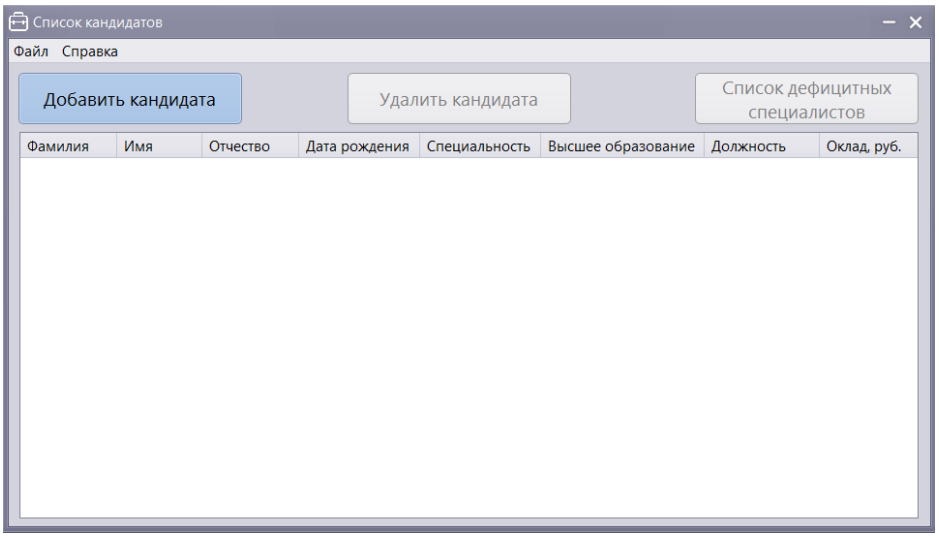

Рисунок 5.3 – Окно просмотра списка с кандидатами

5.1.4 Окно изменения информации о вакансии

Окно для редактирования вакансии содержит несколько полей, которые необходимо заполнить. У каждого текстового поля присутствуют подсказки для верного ввода. Кнопка «Сохранить» становится доступной после заполнения всех полей корректными данными. Полная информация о вакансии должна быть уникальной. Нажатие на кнопку «Отмена» закрывает окно. Внешний вид окна изображен на рисунке 5.4.

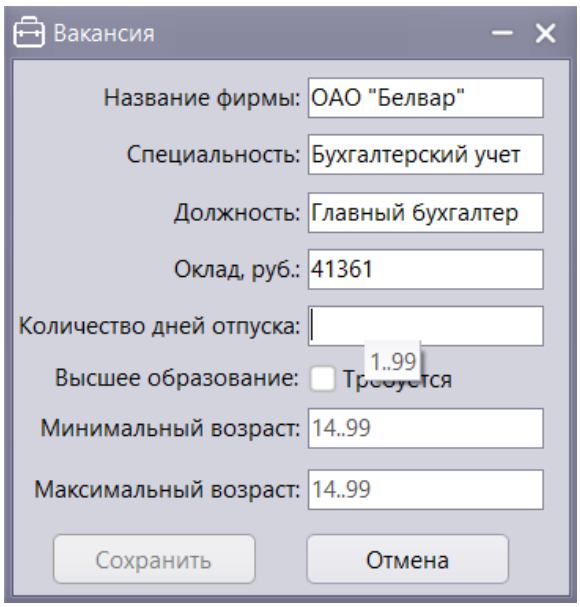

Рисунок 5.4 – Окно изменения информации о вакансии

5.1.5 Окно изменения информации о кандидате

Окно изменения информации о кандидате содержит несколько полей. Внизу окна находятся две кнопки: «Сохранить», обновляющая информацию о кандидате в списке, и «Отмена», закрывающая окно. Кнопка «Сохранить» становится активной после заполнения всех полей. Полная информация о кандидате должна быть уникальной. Внешний вид данного окна представлен на рисунке 5.5.

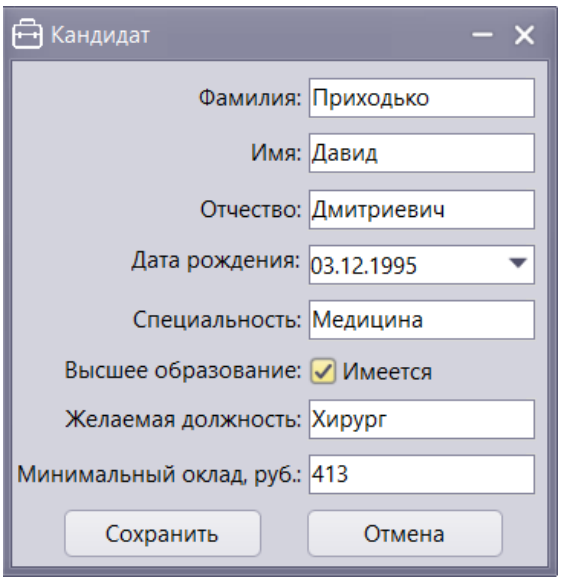

Рисунок 5.5 – Окно изменения информации о кандидате

5.1.6 Окно подбора кандидатов по вакансии

В верхней части окна находится меню, с помощью которого можно сохранить список в файл. Окно содержит две таблицы: верхняя отображает вакансию, по которой подбираются кандидаты; нижняя используется для показа найденных кандидатов. Внешний вид окна представлен на рисунке 5.6.

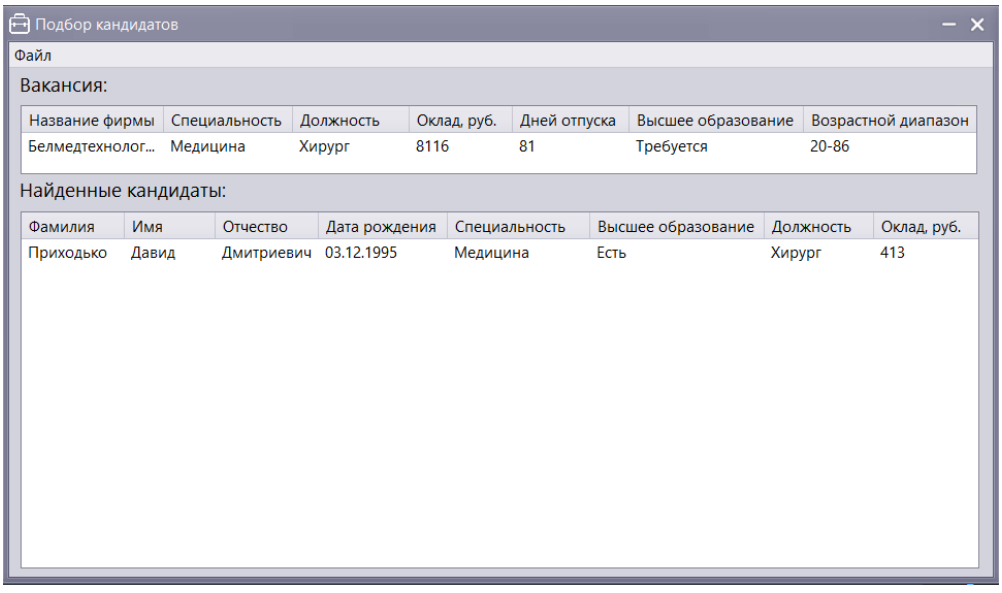

Рисунок 5.6 – Окно подбора кандидатов по вакансии

5.1.7 Окно просмотра списка с дефицитными специалистами

В верхней части окна расположено меню, через которое доступна функция сохранения списка дефицитных специалистов в файл. Ниже

расположена таблица для отображения списка. Внешний вид окна просмотра списка дефицитных специалистов представлен на рисунке 5.7.

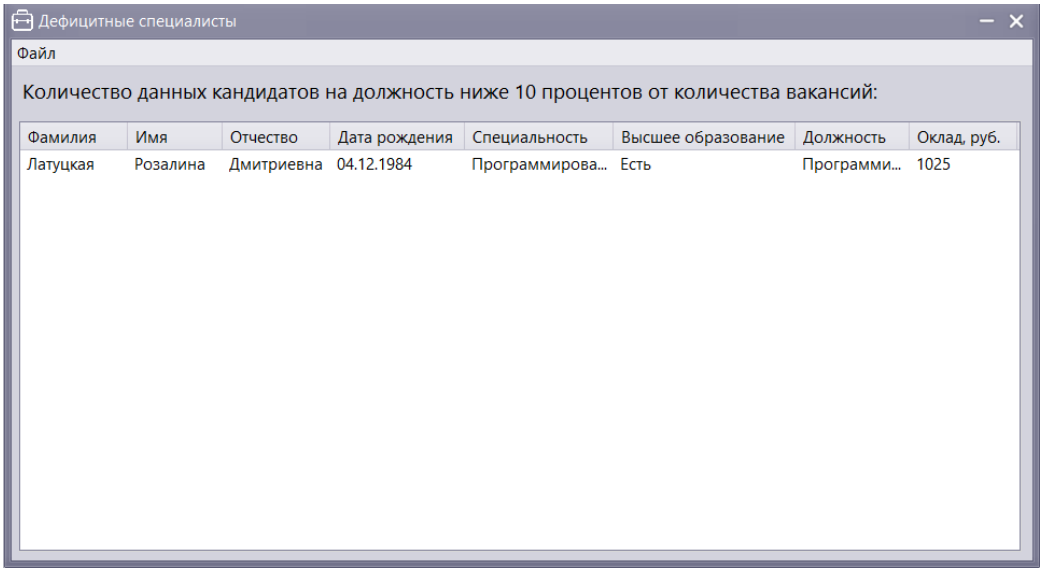

Рисунок 5.7 – Окно просмотра списка с дефицитными специалистами

5.2 Управление программным средством

Для начала работы над одним из списков требуется нажать на соответствующую кнопку в главном меню. Для добавления и удаления записей используются кнопки. Перед удалением необходимо выбрать необходимую запись в таблице. Для редактирования нужно дважды нажать на нужный столбец таблицы.

Ввод данных осуществляется через специальные окна программного средства, либо через типизированные файлы с определенным форматом.

Выходными данными являются основные списки вакансий и кандидатов, а также списки с подобранными кандидатами по вакансии и с дефицитными специалистами.

Вывод данных доступен в двух форматах: в соответствующие таблицы визуального приложения, а также в типизированные файлы.

ЗАКЛЮЧЕНИЕ

В настоящее время приложения по поиску вакансий набирают все большую популярность. Для специалистов они позволяют облегчить процесс поиска работы, а для компаний – автоматизировать и упростить нахождение работников по выбранным критериям.

В рамках данной учебной практики было разработано программное средство «Биржа труда», которое обеспечивает возможность хранения данных о списках с вакансиями и кандидатами, а также осуществляет взаимодействие между этими списками. Данное приложение позволяет пользователю создавать и управлять списками вакансий и кандидатов, подбирать кандидатов, просматривать дефицитных специалистов. Согласно поставленным задачам, при разработке были реализованы следующие функции:

– подбор возможных кандидатов по каждой вакансии;
– формирование списка дефицитных специалистов;
– добавление, удаление и корректирование записей списков;
– просмотр списков целиком.

Для успешного достижения всех поставленных целей при разработке потребовалось изучить графические возможности языка Delphi. В результате был создан простой и понятный пользователю интерфейс, позволяющий отображать содержимое списков и проводить над ними работу.

В дальнейшем приложение планируется улучшить путем добавления поиска по вакансиям и кандидатам, а также усовершенствования уже существующего функционала.

Таким образом, разработанное программное средство является эффективным инструментом для управления данными, связанными с трудоустройством.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Григорьев А.Б. О чем не пишут в книгах по Delphi [Текст]. – СПб.: БХВ-Петербург, 2010. – 576 с.

2. Парижский С.М. Delphi. Учимся на примерах /Под ред. Ю. А. Шпака [Текст]. – Киев: МК-Пресс, 2005. – 216 с.

3. Емельянов, В.И. Основы программирования на Delphi. / В.И. Емельянов. – М.: Высшая школа, 2005. - 231 с.

4. Фаронов В. Delphi. Программирование на языке высокого уровня [Текст]: Учебник для вузов. – СПб.: Питер, 2009. – 640 с.

5. Фаулер М. Предметно-ориентированные языки программирования / [Текст]. – М.: Вильямс, 2011. – 576 с.

6. Фленов М.Е. Библия Delphi [Текст]. 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008. – 800 с.

7. Вальвачев А.Н., Сурков К.А., Сурков Д.А., Четырько Ю.М. Объектно-ориентированное программирование на языках Delphi и C++: учебное пособие для студентов [Электронный ресурс]. – Минск: БГУИР, 2016 – Режим доступа: https://libeldoc.bsuir.by/handle/123456789/8507.

8. RAD Studio Product Documentation – Embarcadero Technologies [Электронный ресурс]. – Электронные данные. – Режим доступа: http://docs.embarcadero.com/products/rad_studio.

ПРИЛОЖЕНИЕ А

Текст программы

```
Unit MainUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Menus,
    Vcl.ComCtrls, Vcl.ExtCtrls, Vcl.Controls;

Type
    TMainForm = Class(TForm)
        MainMenu: TMainMenu;
        MMHelp: TMenuItem;
        LabelProgramName: TLabel;
        ButtonVacancyList: TButton;
        ButtonCandidateList: TButton;
        ButtonExit: TButton;
        MMInstruction: TMenuItem;
        MMSeparator: TMenuItem;
        MMProgramInfo: TMenuItem;
        Procedure ButtonVacancyListClick(Sender: TObject);
        Procedure ButtonCandidateListClick(Sender: TObject);
        Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);
        Procedure ButtonExitClick(Sender: TObject);
        Procedure MMInstructionClick(Sender: TObject);
        Procedure MMProgramInfoClick(Sender: TObject);
        Procedure FormResize(Sender: TObject);
    End;

Const
    MAXLEN = 20;
    MINSALARY = 1;
    MAXSALARY = 99_999;
    MINVACATION = 1;
    MAXVACATION = 99;
    MINWORKAGE = 14;
    MAXWORKAGE = 99;
    MAXRECORDAMOUNT = 100;
    CANDIDATEFILEEXT = '.can';
    VACANCYFILEEXT = '.vac';
    NULL = #0;
    BACKSPACE = #8;

Function IsFileExtCorrect(Path: String; Const EXT: String): Boolean;
Procedure ClearListView(ListView: TListView);
Procedure StrEditKeyPress(LEdit: TLabeledEdit; Var Key: Char; Const MAXLENGTH:
                          Integer);
Procedure IntEditKeyPress(LEdit: TLabeledEdit; Var Key: Char; Const MAX: Integer);
Function IsNumCorrect(Value: Integer; Const MINVALUE, MAXVALUE: Integer): Boolean;
Function IsStrEditCorrect(LEdit: TLabeledEdit): Boolean;
Function IsIntEditCorrect(LEdit: TLabeledEdit; Const MINVALUE, MAXVALUE: Integer):
                          Boolean;
Procedure ShowInstruction();
Procedure ShowProgramInfo();
```

```
Var
    MainForm: TMainForm;

Implementation

{$R *.dfm}

Uses VacancyListUnit, CandidateListUnit;

Function IsStrEditCorrect(LEdit: TLabeledEdit): Boolean;
Begin
    IsStrEditCorrect := (Length(LEdit.Text) > 0) And (Length(LEdit.Text) <
                         MAXLEN + 1);
End;

Function IsNumCorrect(Value: Integer; Const MINVALUE, MAXVALUE: Integer): Boolean;
Begin
    IsNumCorrect := (Value > MINVALUE - 1) And (Value < MAXVALUE + 1);
End;

Function IsIntEditCorrect(LEdit: TLabeledEdit; Const MINVALUE, MAXVALUE: Integer):
                          Boolean;
Var
    Value: Integer;
Begin
    IsIntEditCorrect := TryStrToInt(LEdit.Text, Value) And IsNumCorrect(Value,
                        MINVALUE, MAXVALUE);
End;

Procedure StrEditKeyPress(LEdit: TLabeledEdit; Var Key: Char; Const MAXLENGTH:
                          Integer);
Begin
    If (Key <> BACKSPACE) And Not(Length(LEdit.Text) < MAXLENGTH) Then
        Key := NULL;
End;

Procedure IntEditKeyPress(LEdit: TLabeledEdit; Var Key: Char; Const MAX: Integer);
Begin
    If (Key <> BACKSPACE) And Not(Length(LEdit.Text) < Length(IntToStr(MAX))) Then
        Key := NULL
    Else
        If (LEdit.SelStart = 0) And (Key = '0') Then
            Key := NULL
        Else
            If (Length(LEdit.Text) > 0) And (StrToInt(LEdit.Text) = 0) And
               (LEdit.SelStart <> 0) And (Key <> BACKSPACE) Then
                Key := NULL;
End;

Procedure ClearListView(ListView: TListView);
Begin
    While ListView.Items.Count <> 0 Do
        ListView.Items[0].Delete;
End;

Procedure TMainForm.ButtonCandidateListClick(Sender: TObject);
Begin
    MainForm.Visible := False;
    CandidateListForm.ShowModal;
End;
```

```
Procedure TMainForm.ButtonExitClick(Sender: TObject);
Begin
    Close;
End;

Procedure TMainForm.ButtonVacancyListClick(Sender: TObject);
Begin
    MainForm.Visible := False;
    VacancyListForm.ShowModal;
End;

Procedure TMainForm.FormCloseQuery(Sender: TObject; Var CanClose: Boolean);
Var
    ButtonSelected: Integer;
Begin
    If Not IsVacancyListSaved Then
    Begin
        ButtonSelected := Application.MessageBox('Вы хотите сохранить изменения в
                                                 списке вакансий?', 'Выход',
                                                 MB_YESNOCANCEL + MB_ICONQUESTION);
        If ButtonSelected = MrYes Then
        Begin
            VacancyListForm.MMSaveFile.Click;
            If Not IsVacancyListSaved Then
                Close;
        End
        Else
            CanClose := IsCandidateListSaved And (ButtonSelected = MrNo);
        IsVacancyListSaved := True;
    End;
    If Not IsCandidateListSaved Then
    Begin
        ButtonSelected := Application.MessageBox('Вы хотите сохранить изменения в
                                                 списке кандидатов?', 'Выход',
                                                 MB_YESNOCANCEL + MB_ICONQUESTION);
        If ButtonSelected = MrYes Then
        Begin
            CandidateListForm.MMSaveFile.Click;
            If Not IsCandidateListSaved Then
                Close;
        End
        Else
            CanClose := ButtonSelected = MrNo;
        IsCandidateListSaved := True;
    End;
End;

Procedure TMainForm.FormResize(Sender: TObject);
Begin
    Left := (Screen.Width - Width) Div 2;
    Top := (Screen.Height - Height) Div 2;
End;

Procedure ShowInstruction();
Begin
    Application.MessageBox
      ('1. Для добавления записи в список нажмите на кнопку "Добавить", и введите
        требующуюся информацию.'#13#10 + '2. Для редактирования записи нажмите
        дважды на нужную строку списка.'#13#10 + '3. Для удаления записи выберите ее
        в списке и нажмите на кнопку "Удалить".'#13#10 + '4. Для подбора кандидатов
```

```
        для вакансии выберите нужную вакансию в списке и нажмите на кнопку "Подобрать
        кандидатов".'#13#10 + '5. Для просмотра дефицитных специалистов нажмите
        соответствующую кнопку в окне списка с кандидатами.'#13#10 + '6. Максимальное
        количество записей в каждом списке - 100.'#13#10 + '7. Формат для файлов с
        вакансиями: *.vac.'#13#10 + '8. Формат для файлов с кандидатами: *.can.',
       'Инструкция', MB_OK);
End;

Procedure ShowProgramInfo();
Begin
    Application.MessageBox('Биржа труда'#13#10 + 'Разработчик: Городко Ксения
                           Евгеньевна, 351005'#13#10 + 'Учебная практика
                           (ознакомительная), вариант 15'#13#10 + 'БГУИР 2024',
                          'О программе', MB_OK);
End;

Procedure TMainForm.MMInstructionClick(Sender: TObject);
Begin
    ShowInstruction();
End;

Procedure TMainForm.MMProgramInfoClick(Sender: TObject);
Begin
    ShowProgramInfo();
End;

Function IsFileExtCorrect(Path: String; Const EXT: String): Boolean;
Begin
    If ExtractFileExt(Path) <> EXT Then
        Application.MessageBox(PWideChar('Файл должен иметь разрешение ' + EXT +
                               '!'), 'Ошибка', MB_ICONERROR);
    IsFileExtCorrect := ExtractFileExt(Path) = EXT;
End;

End.
```

```
Unit VacancyListUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls,
    Vcl.Menus, Vcl.StdCtrls, MainUnit, Vcl.ExtCtrls, Vcl.DBCtrls;

Type
    PVacancy = ^TVacancy;
    TVacancyInfo = Record
        FirmName, Speciality, Title: String[MAXLEN];
        Salary, VacationDays: Integer;
        IsHighEducationRequired: Boolean;
        MinAge, MaxAge: Integer;
    End;
    TVacancy = Record
        Info: TVacancyInfo;
        Next: PVacancy;
    End;
    TVacancyListForm = Class(TForm)
        MainMenu: TMainMenu;
        MMFile: TMenuItem;
        MMOpenFile: TMenuItem;
        MMSaveFile: TMenuItem;
        ListView: TListView;
        ButtonAdd: TButton;
        ButtonDelete: TButton;
        ButtonFindCandidates: TButton;
        SaveDialog: TSaveDialog;
        OpenDialog: TOpenDialog;
        MMHelp: TMenuItem;
        MMProgramInfo: TMenuItem;
        MMSeparator: TMenuItem;
        MMInstruction: TMenuItem;
        Procedure ButtonAddClick(Sender: TObject);
        Procedure ListViewSelectItem(Sender: TObject; Item: TListItem; Selected:
                                     Boolean);
        Procedure ListViewChange(Sender: TObject; Item: TListItem; Change:
                                 TItemChange);
        Procedure ButtonDeleteClick(Sender: TObject);
        Procedure ListViewDblClick(Sender: TObject);
        Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);
        Procedure MMSaveFileClick(Sender: TObject);
        Procedure MMOpenFileClick(Sender: TObject);
        Procedure ButtonFindCandidatesClick(Sender: TObject);
        Procedure FormClose(Sender: TObject; Var Action: TCloseAction);
        Procedure MMInstructionClick(Sender: TObject);
        Procedure MMProgramInfoClick(Sender: TObject);
        Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):
                          Boolean;
        Procedure ListViewDeletion(Sender: TObject; Item: TListItem);
        Procedure ReadVacancyListFromFile();
        Procedure FormCreate(Sender: TObject);
    End;

Procedure AddVacancyToListView(VacancyInfo: TVacancyInfo; ListView: TListView);
Procedure AddVacancy(VacancyInfo: TVacancyInfo; Var Head: PVacancy);
Procedure EditVacancy(OldInfo, NewInfo: TVacancyInfo);
```

```
Function IsVacancyInList(Info: TVacancyInfo): Boolean;

Var
    VacancyListForm: TVacancyListForm;
    VacancyHead: PVacancy;
    IsVacancyListSaved: Boolean;
    VacancyAmount: Integer;

Implementation

{$R *.dfm}

Uses VacancyUnit, CandidateSelectUnit;

Const
    HIGHEDUCATIONREQUIRED: Array [Boolean] Of String = ('Не требуется', 'Требуется');

Procedure AddVacancyToListView(VacancyInfo: TVacancyInfo; ListView: TListView);
Var
    NewItem: TListItem;
Begin
    NewItem := ListView.Items.Add;
    NewItem.Caption := String(VacancyInfo.FirmName);
    With NewItem.SubItems, VacancyInfo Do
    Begin
        Add(String(Speciality));
        Add(String(Title));
        Add(IntToStr(Salary));
        Add(IntToStr(VacationDays));
        Add(HIGHEDUCATIONREQUIRED[IsHighEducationRequired]);
        Add(IntToStr(MinAge) + '-' + IntToStr(MaxAge));
    End;
    IsVacancyListSaved := False;
End;

Procedure AddVacancy(VacancyInfo: TVacancyInfo; Var Head: PVacancy);
Var
    NewVacancy, Temp: PVacancy;
Begin
    NewVacancy := New(PVacancy);
    NewVacancy^.Info := VacancyInfo;
    NewVacancy^.Next := Nil;
    If Head = Nil Then
        Head := NewVacancy
    Else
    Begin
        Temp := Head;
        While Temp^.Next <> Nil Do
            Temp := Temp^.Next;
        Temp^.Next := NewVacancy;
    End;
End;

Function AreVacanciesEqual(Vacancy1, Vacancy2: TVacancyInfo): Boolean;
Begin
    AreVacanciesEqual := (Vacancy1.FirmName = Vacancy2.FirmName) And
                         (Vacancy1.Speciality = Vacancy2.Speciality) And
                         (Vacancy1.Title = Vacancy2.Title) And (Vacancy1.Salary =
                         Vacancy2.Salary) And (Vacancy1.VacationDays =
                         Vacancy2.VacationDays) And (Vacancy1.
```

```
                        IsHighEducationRequired = Vacancy2.IsHighEducationRequired)
                        And (Vacancy1.MinAge = Vacancy2.MinAge) And
                        (Vacancy1.MaxAge = Vacancy2.MaxAge);
End;

Procedure EditVacancyInListView(NewInfo: TVacancyInfo);
Begin
    With VacancyListForm.ListView.Selected, NewInfo Do
    Begin
        Caption := String(FirmName);
        SubItems[0] := String(Speciality);
        SubItems[1] := String(Title);
        SubItems[2] := IntToStr(Salary);
        SubItems[3] := IntToStr(VacationDays);
        SubItems[4] := HIGHEDUCATIONREQUIRED[IsHighEducationRequired];
        SubItems[5] := IntToStr(MinAge) + '-' + IntToStr(MaxAge);
    End;
    IsVacancyListSaved := False;
End;

Function IsVacancyInList(Info: TVacancyInfo): Boolean;
Var
    Temp: PVacancy;
Begin
    Temp := VacancyHead;
    While (Temp <> Nil) And Not AreVacanciesEqual(Info, Temp^.Info) Do
        Temp := Temp.Next;
    IsVacancyInList := Temp <> Nil;
End;

Procedure EditVacancy(OldInfo, NewInfo: TVacancyInfo);
Var
    Temp: PVacancy;
Begin
    Temp := VacancyHead;
    While Not AreVacanciesEqual(OldInfo, Temp^.Info) Do
        Temp := Temp^.Next;
    Temp^.Info := NewInfo;
    EditVacancyInListView(NewInfo);
End;

Procedure TVacancyListForm.FormClose(Sender: TObject; Var Action: TCloseAction);
Begin
    MainForm.Visible := True;
End;

Procedure TVacancyListForm.FormCreate(Sender: TObject);
Begin
    VacancyHead := Nil;
    IsVacancyListSaved := True;
    VacancyAmount := 0;
End;

Function TVacancyListForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp:
                                   Boolean): Boolean;
Begin
    ShowInstruction();
    CallHelp := False;
End;
Procedure TVacancyListForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:
```

```
                                        TShiftState);
Begin
    If Key = VK_ESCAPE Then
        Close
    Else
        If (Key = VK_DELETE) And ButtonDelete.Enabled Then
            ButtonDelete.Click;
End;

Procedure DeleteVacancy(VacancyInfo: TVacancyInfo);
Var
    Temp, Curr: PVacancy;
Begin
    Temp := VacancyHead;
    If AreVacanciesEqual(VacancyInfo, Temp^.Info) Then
        VacancyHead := Temp^.Next
    Else
    Begin
        While Not AreVacanciesEqual(VacancyInfo, Temp^.Next^.Info) Do
            Temp := Temp^.Next;
        Curr := Temp;
        Temp := Temp^.Next;
        Curr^.Next := Curr^.Next^.Next;
    End;
    Dispose(Temp);
End;

Procedure DeleteVacancyList(Var Head: PVacancy);
Var
    Curr: PVacancy;
Begin
    Curr := Head;
    While Curr <> Nil Do
    Begin
        Head := Head^.Next;
        Dispose(Curr);
        Curr := Head;
    End;
End;

Procedure TVacancyListForm.ButtonAddClick(Sender: TObject);
Begin
    If VacancyAmount < MAXRECORDAMOUNT Then
        VacancyForm.ShowModal
    Else
        Application.MessageBox('Достигнуто максимальное число вакансий!',
                               'Ошибка', MB_ICONERROR);
End;

Function GetVacancyInfo(Item: TListItem): TVacancyInfo;
Var
    Vacancy: TVacancyInfo;
Begin
    With Vacancy, Item Do
    Begin
        FirmName := ShortString(Caption);
        Speciality := ShortString(SubItems[0]);
        Title := ShortString(SubItems[1]);
        Salary := StrToInt(SubItems[2]);
        VacationDays := StrToInt(SubItems[3]);
```

```
        IsHighEducationRequired := SubItems[4] = HIGHEDUCATIONREQUIRED[True];
        MinAge := StrToInt(Copy(SubItems[5], 1, Pos('-', SubItems[5]) - 1));
        MaxAge := StrToInt(Copy(SubItems[5], Pos('-', SubItems[5]) + 1, 2));
    End;
    GetVacancyInfo := Vacancy;
End;

Procedure TVacancyListForm.ButtonDeleteClick(Sender: TObject);
Var
    ButtonSelected: Integer;
Begin
    ButtonSelected := Application.MessageBox('Вы уверены, что хотите удалить
                                             выделенную вакансию?', 'Удаление',
                                             MB_YESNO + MB_ICONQUESTION);
    If ButtonSelected = MrYes Then
    Begin
        DeleteVacancy(GetVacancyInfo(ListView.Selected));
        ListView.Selected.Delete;
        Dec(VacancyAmount);
    End
    Else
        ListView.ClearSelection;
End;

Procedure TVacancyListForm.ButtonFindCandidatesClick(Sender: TObject);
Begin
    Vacancy := GetVacancyInfo(ListView.Selected);
    CandidateSelectForm.ShowModal;
End;

Procedure TVacancyListForm.ListViewChange(Sender: TObject; Item: TListItem;
                                          Change: TItemChange);
Begin
    MMSaveFile.Enabled := ListView.Items.Count > 0;
End;

Procedure TVacancyListForm.ListViewDblClick(Sender: TObject);
Begin
    If ListView.Selected <> Nil Then
    Begin
        OldInfo := GetVacancyInfo(ListView.Selected);
        IsEditing := True;
        VacancyForm.ShowModal;
    End;
End;

Procedure TVacancyListForm.ListViewDeletion(Sender: TObject; Item: TListItem);
Begin
    MMSaveFile.Enabled := ListView.Items.Count > 1;
    IsVacancyListSaved := ListView.Items.Count = 1;
End;

Procedure TVacancyListForm.ListViewSelectItem(Sender: TObject; Item: TListItem;
                                              Selected: Boolean);
Begin
    ButtonDelete.Enabled := Selected;
    ButtonFindCandidates.Enabled := Selected;
End;

Function IsVacancyCorrect(Vacancy: TVacancyInfo): Boolean;
```

```
Begin
    IsVacancyCorrect := IsNumCorrect(Vacancy.Salary, MINSALARY, MAXSALARY) And
                        IsNumCorrect(Vacancy.VacationDays, MINVACATION, MAXVACATION)
                        And IsNumCorrect(Vacancy.MinAge, MINWORKAGE, MAXWORKAGE) And
                        IsNumCorrect(Vacancy.MaxAge, MINWORKAGE, MAXWORKAGE) And
                        Not(Vacancy.MinAge > Vacancy.MaxAge);
End;

Procedure TVacancyListForm.ReadVacancyListFromFile();
Var
    InputFile: File Of TVacancyInfo;
    VacancyInfo: TVacancyInfo;
    Head: PVacancy;
    IsCorrect: Boolean;
    Count: Integer;
Begin
    IsCorrect := OpenDialog.Execute And IsFileExtCorrect(OpenDialog.FileName,
                                                         VACANCYFILEEXT);
    If IsCorrect Then
    Begin
        Try
            Head := Nil;
            Count := 0;
            AssignFile(InputFile, OpenDialog.FileName);
            Try
                Reset(InputFile);
                If FileSize(InputFile) > MAXRECORDAMOUNT Then
                    IsCorrect := False;
                While Not Eof(InputFile) And IsCorrect Do
                Begin
                    Read(InputFile, VacancyInfo);
                    AddVacancy(VacancyInfo, Head);
                    IsCorrect := IsVacancyCorrect(VacancyInfo);
                    Inc(Count);
                End;
            Except
                IsCorrect := False;
            End;
        Finally
            CloseFile(InputFile);
        End;
        If IsCorrect Then
        Begin
            DeleteVacancyList(VacancyHead);
            ClearListView(ListView);
            VacancyHead := Head;
            While Head <> Nil Do
            Begin
                AddVacancyToListView(Head^.Info, ListView);
                Head := Head^.Next;
            End;
            VacancyAmount := Count;
            IsVacancyListSaved := True;
        End
        Else
        Begin
            DeleteVacancyList(Head);
            Application.MessageBox('Произошла ошибка при открытии файла! Проверьте
                                   корректность данных!', 'Ошибка', MB_ICONERROR);
        End;
```

```
    End;
End;

Procedure TVacancyListForm.MMOpenFileClick(Sender: TObject);
Var
    ButtonSelected: Integer;
Begin
    If Not IsVacancyListSaved Then
    Begin
        ButtonSelected := Application.MessageBox('Вы хотите сохранить изменения в
                                                  списке вакансий?', 'Выход',
                                                  MB_YESNOCANCEL + MB_ICONQUESTION);
        If ButtonSelected = MrYes Then
            MMSaveFile.Click;
    End;
    ReadVacancyListFromFile();
End;

Procedure SaveVacancyListToFile(Path: String);
Var
    OutputFile: File Of TVacancyInfo;
    Temp: PVacancy;
Begin
    Try
        Temp := VacancyHead;
        AssignFile(OutputFile, Path);
        Rewrite(OutputFile);
        While Temp <> Nil Do
        Begin
            Write(OutputFile, Temp^.Info);
            Temp := Temp^.Next;
        End;
    Finally
        CloseFile(OutputFile);
    End;
End;

Procedure TVacancyListForm.MMSaveFileClick(Sender: TObject);
Var
    IsCorrect: Boolean;
Begin
    IsCorrect := SaveDialog.Execute And IsFileExtCorrect(SaveDialog.FileName,
                                                         VACANCYFILEEXT);
    If IsCorrect Then
        SaveVacancyListToFile(SaveDialog.FileName);
    IsVacancyListSaved := IsCorrect;
End;

Procedure TVacancyListForm.MMProgramInfoClick(Sender: TObject);
Begin
    ShowProgramInfo();
End;

Procedure TVacancyListForm.MMInstructionClick(Sender: TObject);
Begin
    ShowInstruction();
End;
End.
```

```
Unit VacancyUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.StdCtrls,
    Vcl.ExtCtrls, VacancyListUnit, Vcl.Mask;

Type
    TVacancyForm = Class(TForm)
        LabelHighEducation: TLabel;
        ButtonSave: TButton;
        ButtonCancel: TButton;
        CheckBoxHighEducation: TCheckBox;
        LEditFirmName: TLabeledEdit;
        LEditSpeciality: TLabeledEdit;
        LEditTitle: TLabeledEdit;
        LEditSalary: TLabeledEdit;
        LEditVacationDays: TLabeledEdit;
        LEditMinAge: TLabeledEdit;
        LEditMaxAge: TLabeledEdit;
        Procedure ButtonCancelClick(Sender: TObject);
        Procedure ControlOnKeyDown(Sender: TObject; Var Key: Word; Shift:
                                   TShiftState);
        Procedure EditOnChange(Sender: TObject);
        Procedure ClearControls();
        Procedure FormClose(Sender: TObject; Var Action: TCloseAction);
        Procedure ButtonSaveClick(Sender: TObject);
        Procedure FormShow(Sender: TObject);
        Procedure LEditFirmNameKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditSpecialityKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditTitleKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditSalaryKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditVacationDaysKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditMinAgeKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditMaxAgeKeyPress(Sender: TObject; Var Key: Char);
        Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);
    End;

Var
    VacancyForm: TVacancyForm;
    IsEditing: Boolean;
    OldInfo: TVacancyInfo;

Implementation

{$R *.dfm}

Uses MainUnit;

Function IsAgeRangeCorrect(): Boolean;
Begin
    With VacancyForm Do
        IsAgeRangeCorrect := IsIntEditCorrect(LEditMinAge, MINWORKAGE,
                             MAXWORKAGE) And IsIntEditCorrect(LEditMaxAge,
                             MINWORKAGE, MAXWORKAGE) And Not(StrToInt
                             (LEditMinAge.Text) > StrToInt(LEditMaxAge.Text));
End;
```

```
Procedure TVacancyForm.EditOnChange(Sender: TObject);
Begin
    ButtonSave.Enabled := IsStrEditCorrect(LEditFirmName) And
                          IsStrEditCorrect(LEditSpeciality) And
                          IsStrEditCorrect(LEditTitle) And
                          IsIntEditCorrect(LEditSalary, MINSALARY, MAXSALARY) And
                          IsIntEditCorrect(LEditVacationDays, MINVACATION,
                          MAXVACATION) And IsAgeRangeCorrect();
End;

Procedure TVacancyForm.ButtonSaveClick(Sender: TObject);
Var
    NewInfo: TVacancyInfo;
Begin
    With NewInfo Do
    Begin
        FirmName := ShortString(LEditFirmName.Text);
        Speciality := ShortString(LEditSpeciality.Text);
        Title := ShortString(LEditTitle.Text);
        Salary := StrToInt(LEditSalary.Text);
        VacationDays := StrToInt(LEditVacationDays.Text);
        IsHighEducationRequired := CheckBoxHighEducation.Checked;
        MinAge := StrToInt(LEditMinAge.Text);
        MaxAge := StrToInt(LEditMaxAge.Text);
    End;
    If IsEditing Then
        EditVacancy(OldInfo, NewInfo)
    Else
    Begin
        If Not IsVacancyInList(NewInfo) Then
        Begin
            AddVacancy(NewInfo, VacancyHead);
            AddVacancyToListView(NewInfo, VacancyListForm.ListView);
            Inc(VacancyAmount);
        End
        Else
            Application.MessageBox('Такая вакансия уже есть в списке!', 'Ошибка',
                                   MB_ICONERROR);
    End;
    Close;
End;

Procedure TVacancyForm.ControlOnKeyDown(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)
End;

Procedure TVacancyForm.ClearControls();
Begin
    LEditFirmName.Text := '';
    LEditSpeciality.Text := '';
    LEditTitle.Text := '';
    LEditSalary.Text := '';
    LEditVacationDays.Text := '';
    CheckBoxHighEducation.Checked := False;
```

```
    LEditMinAge.Text := '';
    LEditMaxAge.Text := '';
End;

Procedure TVacancyForm.ButtonCancelClick(Sender: TObject);
Begin
    Close;
End;

Procedure TVacancyForm.FormClose(Sender: TObject; Var Action: TCloseAction);
Begin
    IsEditing := False;
    ClearControls();
    LEditFirmName.SetFocus;
End;

Procedure TVacancyForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift:
                                   TShiftState);
Begin
    If Key = VK_ESCAPE Then
        Close
    Else
        If (Key = 13) And ButtonSave.Enabled Then
            ButtonSave.Click;
End;

Procedure TVacancyForm.FormShow(Sender: TObject);
Begin
    If IsEditing Then
        With OldInfo Do
        Begin
            LEditFirmName.Text := String(FirmName);
            LEditSpeciality.Text := String(Speciality);
            LEditTitle.Text := String(Title);
            LEditSalary.Text := IntToStr(Salary);
            LEditVacationDays.Text := IntToStr(VacationDays);
            CheckBoxHighEducation.Checked := IsHighEducationRequired;
            LEditMinAge.Text := IntToStr(MinAge);
            LEditMaxAge.Text := IntToStr(MaxAge);
        End;
End;

Procedure TVacancyForm.LEditFirmNameKeyPress(Sender: TObject; Var Key: Char);
Begin
    StrEditKeyPress(LEditFirmName, Key, MAXLEN);
End;

Procedure TVacancyForm.LEditMaxAgeKeyPress(Sender: TObject; Var Key: Char);
Begin
    IntEditKeyPress(LEditMaxAge, Key, MAXWORKAGE);
End;

Procedure TVacancyForm.LEditMinAgeKeyPress(Sender: TObject; Var Key: Char);
Begin
    IntEditKeyPress(LEditMinAge, Key, MAXWORKAGE);
End;

Procedure TVacancyForm.LEditSalaryKeyPress(Sender: TObject; Var Key: Char);
Begin
    IntEditKeyPress(LEditSalary, Key, MAXSALARY);
```

```
End;

Procedure TVacancyForm.LEditSpecialityKeyPress(Sender: TObject; Var Key: Char);
Begin
    StrEditKeyPress(LEditSpeciality, Key, MAXLEN);
End;

Procedure TVacancyForm.LEditTitleKeyPress(Sender: TObject; Var Key: Char);
Begin
    StrEditKeyPress(LEditTitle, Key, MAXLEN);
End;

Procedure TVacancyForm.LEditVacationDaysKeyPress(Sender: TObject; Var Key: Char);
Begin
    IntEditKeyPress(LEditVacationDays, Key, MAXVACATION);
End;

End.
```

```
Unit CandidateListUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
    Vcl.Menus, Vcl.ComCtrls, Vcl.StdCtrls, MainUnit;

Type
    PCandidate = ^TCandidate;
    TCandidateInfo = Record
        Surname, Name, Patronymic, Speciality, Title: String[MAXLEN];
        BirthDate: TDateTime;
        HasHighEducation: Boolean;
        Salary: Integer;
    End;
    TCandidate = Record
        Info: TCandidateInfo;
        Next: PCandidate;
    End;
    TCandidateListForm = Class(TForm)
        ButtonAdd: TButton;
        ButtonDelete: TButton;
        ButtonDeficite: TButton;
        ListView: TListView;
        MainMenu: TMainMenu;
        MMFile: TMenuItem;
        MMOpenFile: TMenuItem;
        MMSaveFile: TMenuItem;
        OpenDialog: TOpenDialog;
        SaveDialog: TSaveDialog;
        MMHelp: TMenuItem;
        MMProgramInfo: TMenuItem;
        MMSeparator: TMenuItem;
        MMInstruction: TMenuItem;
        Procedure ButtonAddClick(Sender: TObject);
        Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);
        Procedure ButtonDeleteClick(Sender: TObject);
        Procedure ListViewChange(Sender: TObject; Item: TListItem; Change:
                                 TItemChange);
        Procedure ListViewDblClick(Sender: TObject);
        Procedure ListViewSelectItem(Sender: TObject; Item: TListItem; Selected:
                                     Boolean);
        Procedure MMOpenFileClick(Sender: TObject);
        Procedure MMSaveFileClick(Sender: TObject);
        Procedure ButtonDeficiteClick(Sender: TObject);
        Procedure FormClose(Sender: TObject; Var Action: TCloseAction);
        Procedure MMInstructionClick(Sender: TObject);
        Procedure MMProgramInfoClick(Sender: TObject);
        Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):
                          Boolean;
        Procedure ListViewDeletion(Sender: TObject; Item: TListItem);
        Procedure FormCreate(Sender: TObject);
        Procedure ReadCandidateListFromFile();
    End;

Procedure AddCandidateToListView(CandidateInfo: TCandidateInfo; ListView: TListView);
Procedure AddCandidate(CandidateInfo: TCandidateInfo; Var Head: PCandidate);
Procedure EditCandidate(OldInfo, NewInfo: TCandidateInfo);
```

```
Procedure DeleteCandidateList(Var Head: PCandidate);
Procedure SaveCandidateListToFile(Head: PCandidate; Path: String);
Function IsCandidateInList(Info: TCandidateInfo): Boolean;

Var
    CandidateListForm: TCandidateListForm;
    CandidateHead: PCandidate;
    IsCandidateListSaved: Boolean;
    CandidateAmount: Integer;

Implementation

{$R *.dfm}

Uses CandidateUnit, DeficitUnit, DateUtils;

Const
    HIGHEDUCATION: Array [Boolean] Of String = ('Нет', 'Есть');

Procedure AddCandidateToListView(CandidateInfo: TCandidateInfo; ListView: TListView);
Var
    NewItem: TListItem;
Begin
    NewItem := ListView.Items.Add;
    NewItem.Caption := String(CandidateInfo.Surname);
    With NewItem.SubItems, CandidateInfo Do
    Begin
        Add(String(Name));
        Add(String(Patronymic));
        Add(DateToStr(BirthDate));
        Add(String(Speciality));
        Add(HIGHEDUCATION[HasHighEducation]);
        Add(String(Title));
        Add(IntToStr(Salary));
    End;
    IsCandidateListSaved := False;
End;

Procedure AddCandidate(CandidateInfo: TCandidateInfo; Var Head: PCandidate);
Var
    NewCandidate, Temp: PCandidate;
Begin
    NewCandidate := New(PCandidate);
    NewCandidate^.Info := CandidateInfo;
    NewCandidate^.Next := Nil;
    If Head = Nil Then
        Head := NewCandidate
    Else
    Begin
        Temp := Head;
        While Temp^.Next <> Nil Do
            Temp := Temp^.Next;
        Temp^.Next := NewCandidate;
    End;
End;

Function AreCandidatesEqual(Candidate1, Candidate2: TCandidateInfo): Boolean;
Begin
    AreCandidatesEqual := (Candidate1.Surname = Candidate2.Surname) And
                          (Candidate1.Name = Candidate2.Name) And
```

```
                            (Candidate1.Patronymic = Candidate2.Patronymic) And
                            (Candidate1.Speciality = Candidate2.Speciality) And
                            (Candidate1.Title = Candidate2.Title) And
                            (Candidate1.BirthDate = Candidate2.BirthDate) And
                            (Candidate1.HasHighEducation = Candidate2.HasHighEducation)
                            And (Candidate1.Salary = Candidate2.Salary);
End;

Procedure EditCandidateInListView(NewInfo: TCandidateInfo);
Begin
    With CandidateListForm.ListView.Selected, NewInfo Do
    Begin
        Caption := String(Surname);
        SubItems[0] := String(Name);
        SubItems[1] := String(Patronymic);
        SubItems[2] := DateToStr(BirthDate);
        SubItems[3] := String(Speciality);
        SubItems[4] := HIGHEDUCATION[HasHighEducation];
        SubItems[5] := String(Title);
        SubItems[6] := IntToStr(Salary);
    End;
    IsCandidateListSaved := False;
End;

Procedure EditCandidate(OldInfo, NewInfo: TCandidateInfo);
Var
    Temp: PCandidate;
Begin
    Temp := CandidateHead;
    While Not AreCandidatesEqual(OldInfo, Temp^.Info) Do
        Temp := Temp^.Next;
    Temp^.Info := NewInfo;
    EditCandidateInListView(NewInfo);
End;

Function IsCandidateInList(Info: TCandidateInfo): Boolean;
Var
    Temp: PCandidate;
Begin
    Temp := CandidateHead;
    While (Temp <> Nil) And Not AreCandidatesEqual(Info, Temp^.Info) Do
        Temp := Temp.Next;
    IsCandidateInList := Temp <> Nil;
End;

Procedure DeleteCandidate(CandidateInfo: TCandidateInfo);
Var
    Temp, Curr: PCandidate;
Begin
    Temp := CandidateHead;
    If Not AreCandidatesEqual(CandidateInfo, Temp^.Info) Then
    Begin
        While Not AreCandidatesEqual(CandidateInfo, Temp^.Next^.Info) Do
            Temp := Temp^.Next;
        Curr := Temp;
        Temp := Temp^.Next;
        Curr^.Next := Curr^.Next^.Next;
    End
    Else
        CandidateHead := Temp^.Next;
```

```
    Dispose(Temp);
End;

Procedure DeleteCandidateList(Var Head: PCandidate);
Var
    Temp: PCandidate;
Begin
    Temp := Head;
    While Temp <> Nil Do
    Begin
        Head := Head^.Next;
        Dispose(Temp);
        Temp := Head;
    End;
End;

Function GetCandidateInfo(Item: TListItem): TCandidateInfo;
Var
    Candidate: TCandidateInfo;
Begin
    With Candidate, Item Do
    Begin
        Surname := ShortString(Caption);
        Name := ShortString(SubItems[0]);
        Patronymic := ShortString(SubItems[1]);
        BirthDate := StrToDate(SubItems[2]);
        Speciality := ShortString(SubItems[3]);
        HasHighEducation := HIGHEDUCATION[True] = SubItems[4];
        Title := ShortString(SubItems[5]);
        Salary := StrToInt(SubItems[6]);
    End;
    GetCandidateInfo := Candidate;
End;

Procedure TCandidateListForm.ButtonAddClick(Sender: TObject);
Begin
    If CandidateAmount < MAXRECORDAMOUNT Then
        CandidateForm.ShowModal
    Else
        Application.MessageBox('Достигнуто максимальное число кандидатов!',
                               'Ошибка', MB_ICONERROR);
End;

Procedure TCandidateListForm.ButtonDeficiteClick(Sender: TObject);
Begin
    DeficitForm.ShowModal;
End;

Procedure TCandidateListForm.ButtonDeleteClick(Sender: TObject);
Var
    ButtonSelected: Integer;
Begin
    ButtonSelected := Application.MessageBox('Вы уверены, что хотите удалить
                                             выделенного кандидата?', 'Удаление',
                                             MB_YESNO + MB_ICONQUESTION);
    If ButtonSelected = MrYes Then
    Begin
        DeleteCandidate(GetCandidateInfo(ListView.Selected));
        ListView.Selected.Delete;
        Dec(CandidateAmount);
```

```
    End
    Else
        ListView.ClearSelection;
End;

Procedure TCandidateListForm.FormClose(Sender: TObject; Var Action: TCloseAction);
Begin
    MainForm.Visible := True;
End;

Procedure TCandidateListForm.FormCreate(Sender: TObject);
Begin
    CandidateHead := Nil;
    IsCandidateListSaved := True;
    CandidateAmount := 0;
End;

Function TCandidateListForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp:
                                     Boolean): Boolean;
Begin
    ShowInstruction();
    CallHelp := False;
End;

Procedure TCandidateListForm.FormKeyDown(Sender: TObject; Var Key: Word;
                                         Shift: TShiftState);
Begin
    If Key = VK_ESCAPE Then
        Close
    Else
        If (Key = VK_DELETE) And ButtonDelete.Enabled Then
            ButtonDelete.Click;
End;

Procedure TCandidateListForm.ListViewChange(Sender: TObject; Item: TListItem;
                                            Change: TItemChange);
Begin
    ButtonDeficite.Enabled := ListView.Items.Count > 0;
    MMSaveFile.Enabled := ListView.Items.Count > 0;
End;

Procedure TCandidateListForm.ListViewDblClick(Sender: TObject);
Begin
    If ListView.Selected <> Nil Then
    Begin
        OldInfo := GetCandidateInfo(ListView.Selected);
        IsEditing := True;
        CandidateForm.ShowModal;
    End;
End;

Procedure TCandidateListForm.ListViewDeletion(Sender: TObject; Item: TListItem);
Begin
    ButtonDeficite.Enabled := ListView.Items.Count > 1;
    MMSaveFile.Enabled := ListView.Items.Count > 1;
    IsCandidateListSaved := ListView.Items.Count = 1;
End;
Procedure TCandidateListForm.ListViewSelectItem(Sender: TObject; Item: TListItem;
                                                Selected: Boolean);
Begin
```

```
    ButtonDelete.Enabled := Selected;
End;

Procedure TCandidateListForm.MMInstructionClick(Sender: TObject);
Begin
    ShowInstruction();
End;

Function IsCandidateCorrect(Candidate: TCandidateInfo): Boolean;
Begin
    IsCandidateCorrect := IsNumCorrect(Candidate.Salary, MINSALARY, MAXSALARY)
                          And IsNumCorrect(YearsBetween(Now, Candidate.BirthDate),
                          MINWORKAGE, MAXWORKAGE);
End;

Procedure TCandidateListForm.ReadCandidateListFromFile();
Var
    InputFile: File Of TCandidateInfo;
    CandidateInfo: TCandidateInfo;
    Head: PCandidate;
    IsCorrect: Boolean;
    Count: Integer;
Begin
    IsCorrect := OpenDialog.Execute And IsFileExtCorrect(OpenDialog.FileName,
                                                         CANDIDATEFILEEXT);
    If IsCorrect Then
    Begin
        Try
            Head := Nil;
            Count := 0;
            AssignFile(InputFile, OpenDialog.FileName);
            Try
                Reset(InputFile);
                If FileSize(InputFile) > MAXRECORDAMOUNT Then
                    IsCorrect := False;
                While Not Eof(InputFile) And IsCorrect Do
                Begin
                    Read(InputFile, CandidateInfo);
                    AddCandidate(CandidateInfo, Head);
                    IsCorrect := IsCandidateCorrect(CandidateInfo);
                    Inc(Count);
                End;
            Except
                IsCorrect := False;
            End;
        Finally
            CloseFile(InputFile);
        End;
        If IsCorrect Then
        Begin
            DeleteCandidateList(CandidateHead);
            ClearListView(ListView);
            CandidateHead := Head;
            While Head <> Nil Do
            Begin
                AddCandidateToListView(Head^.Info, ListView);
                Head := Head^.Next;
            End;
            CandidateAmount := Count;
            IsCandidateListSaved := True;
```

```
        End
        Else
        Begin
            DeleteCandidateList(Head);
            Application.MessageBox('Произошла ошибка при открытии файла! Проверьте
                                   корректность данных!', 'Ошибка', MB_ICONERROR);
        End;
    End;
End;

Procedure TCandidateListForm.MMOpenFileClick(Sender: TObject);
Var
    ButtonSelected: Integer;
Begin
    If Not IsCandidateListSaved Then
    Begin
        ButtonSelected := Application.MessageBox('Вы хотите сохранить изменения в
                                                 списке кандидатов?', 'Выход',
                                                 MB_YESNOCANCEL + MB_ICONQUESTION);
        If ButtonSelected = MrYes Then
            MMSaveFile.Click;
    End;
    ReadCandidateListFromFile();
End;

Procedure TCandidateListForm.MMProgramInfoClick(Sender: TObject);
Begin
    ShowProgramInfo();
End;

Procedure SaveCandidateListToFile(Head: PCandidate; Path: String);
Var
    OutputFile: File Of TCandidateInfo;
    Temp: PCandidate;
Begin
    Try
        Temp := Head;
        AssignFile(OutputFile, Path);
        Rewrite(OutputFile);
        While Temp <> Nil Do
        Begin
            Write(OutputFile, Temp^.Info);
            Temp := Temp^.Next;
        End;
    Finally
        CloseFile(OutputFile);
    End;
End;

Procedure TCandidateListForm.MMSaveFileClick(Sender: TObject);
Var
    IsCorrect: Boolean;
Begin
    IsCorrect := SaveDialog.Execute And IsFileExtCorrect(SaveDialog.FileName,
                                                         CANDIDATEFILEEXT);
    If IsCorrect Then
        SaveCandidateListToFile(CandidateHead, SaveDialog.FileName);
    IsCandidateListSaved := IsCorrect;
End;
End.
```

```
Unit CandidateUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.StdCtrls,
    Vcl.ComCtrls, Vcl.ExtCtrls, Vcl.Mask, CandidateListUnit;

Type
    TCandidateForm = Class(TForm)
        LabelHighEducation: TLabel;
        LabelBirthdate: TLabel;
        ButtonSave: TButton;
        ButtonCancel: TButton;
        CheckBoxHighEducation: TCheckBox;
        DateTimePicker: TDateTimePicker;
        LEditSurname: TLabeledEdit;
        LEditName: TLabeledEdit;
        LEditPatronymic: TLabeledEdit;
        LEditSpeciality: TLabeledEdit;
        LEditTitle: TLabeledEdit;
        LEditSalary: TLabeledEdit;
        Procedure ButtonCancelClick(Sender: TObject);
        Procedure EditOnChange(Sender: TObject);
        Procedure ButtonSaveClick(Sender: TObject);
        Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);
        Procedure FormCreate(Sender: TObject);
        Procedure LEditSurnameKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditNameKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditPatronymicKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditSpecialityKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditTitleKeyPress(Sender: TObject; Var Key: Char);
        Procedure LEditSalaryKeyPress(Sender: TObject; Var Key: Char);
        Procedure FormShow(Sender: TObject);
        Procedure FormClose(Sender: TObject; Var Action: TCloseAction);
        Procedure ClearControls();
        Procedure ControlOnKeyDown(Sender: TObject; Var Key: Word; Shift:
                                   TShiftState);
    End;

Var
    CandidateForm: TCandidateForm;
    IsEditing: Boolean;
    OldInfo: TCandidateInfo;

Implementation

{$R *.dfm}

Uses MainUnit;

Procedure TCandidateForm.ButtonSaveClick(Sender: TObject);
Var
    NewInfo: TCandidateInfo;
Begin
    With NewInfo Do
    Begin
        Surname := ShortString(LEditSurname.Text);
        Name := ShortString(LEditName.Text);
```

```
        Patronymic := ShortString(LEditPatronymic.Text);
        Speciality := ShortString(LEditSpeciality.Text);
        BirthDate := DateTimePicker.Date;
        HasHighEducation := CheckBoxHighEducation.Checked;
        Title := ShortString(LEditTitle.Text);
        Salary := StrToInt(LEditSalary.Text);
    End;
    If IsEditing Then
        EditCandidate(OldInfo, NewInfo)
    Else
    Begin
        If Not IsCandidateInList(NewInfo) Then
        Begin
            AddCandidate(NewInfo, CandidateHead);
            AddCandidateToListView(NewInfo, CandidateListForm.ListView);
            Inc(CandidateAmount);
        End
        Else
            Application.MessageBox('Такой кандидат уже есть в списке!',
                                   'Ошибка', MB_ICONERROR);
    End;
    Close;
End;

Procedure TCandidateForm.EditOnChange(Sender: TObject);
Begin
    ButtonSave.Enabled := IsStrEditCorrect(LEditSurname) And
                          IsStrEditCorrect(LEditName) And
                          IsStrEditCorrect(LEditPatronymic) And
                          IsStrEditCorrect(LEditSpeciality) And
                          IsStrEditCorrect(LEditTitle) And
                          IsIntEditCorrect(LEditSalary, MINSALARY, MAXSALARY);
End;

Procedure TCandidateForm.ControlOnKeyDown(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)
End;

Procedure TCandidateForm.ClearControls();
Begin
    LEditSurname.Text := '';
    LEditName.Text := '';
    LEditPatronymic.Text := '';
    DateTimePicker.DateTime := DateTimePicker.MaxDate;
    LEditSpeciality.Text := '';
    CheckBoxHighEducation.Checked := False;
    LEditTitle.Text := '';
    LEditSalary.Text := '';
End;

Procedure TCandidateForm.FormClose(Sender: TObject; Var Action: TCloseAction);
Begin
    IsEditing := False;
    ClearControls();
```

```
    LEditSurname.SetFocus;
End;

Procedure TCandidateForm.FormCreate(Sender: TObject);
Var
    Year, Month, Day: Word;
Begin
    DecodeDate(Now, Year, Month, Day);
    DateTimePicker.MaxDate := EncodeDate(Year - MINWORKAGE, Month, Day);
    DateTimePicker.MinDate := EncodeDate(Year - MAXWORKAGE - 1, Month, Day + 1);
End;

Procedure TCandidateForm.FormKeyDown(Sender: TObject; Var Key: Word;
                                     Shift: TShiftState);
Begin
    If Key = VK_ESCAPE Then
        Close
    Else
        If (Key = 13) And ButtonSave.Enabled Then
            ButtonSave.Click;
End;

Procedure TCandidateForm.FormShow(Sender: TObject);
Begin
    If IsEditing Then
        With OldInfo Do
        Begin
            LEditSurname.Text := String(Surname);
            LEditName.Text := String(Name);
            LEditPatronymic.Text := String(Patronymic);
            DateTimePicker.DateTime := BirthDate;
            LEditSpeciality.Text := String(Speciality);
            CheckBoxHighEducation.Checked := HasHighEducation;
            LEditTitle.Text := String(Title);
            LEditSalary.Text := IntToStr(Salary);
        End;
End;

Procedure TCandidateForm.LEditNameKeyPress(Sender: TObject; Var Key: Char);
Begin
    StrEditKeyPress(LEditName, Key, MAXLEN);
End;

Procedure TCandidateForm.LEditPatronymicKeyPress(Sender: TObject;
  Var Key: Char);
Begin
    StrEditKeyPress(LEditPatronymic, Key, MAXLEN);
End;

Procedure TCandidateForm.LEditSalaryKeyPress(Sender: TObject; Var Key: Char);
Begin
    IntEditKeyPress(LEditSalary, Key, MAXSALARY);
End;

Procedure TCandidateForm.LEditSpecialityKeyPress(Sender: TObject;
  Var Key: Char);
Begin
    StrEditKeyPress(LEditSpeciality, Key, MAXLEN);
End;
```

```
Procedure TCandidateForm.LEditSurnameKeyPress(Sender: TObject; Var Key: Char);
Begin
    StrEditKeyPress(LEditSurname, Key, MAXLEN);
End;

Procedure TCandidateForm.LEditTitleKeyPress(Sender: TObject; Var Key: Char);
Begin
    StrEditKeyPress(LEditTitle, Key, MAXLEN);
End;

Procedure TCandidateForm.ButtonCancelClick(Sender: TObject);
Begin
    Close;
End;

End.
```

```
Unit CandidateSelectUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
    Vcl.Menus, Vcl.ComCtrls, CandidateListUnit, VacancyListUnit;

Type
    TCandidateSelectForm = Class(TForm)
        LabelVacancy: TLabel;
        MainMenu: TMainMenu;
        MMFile: TMenuItem;
        MMSaveFile: TMenuItem;
        ListViewVacancy: TListView;
        LabelCandidates: TLabel;
        ListViewCandidates: TListView;
        SaveDialog: TSaveDialog;
        Procedure FormShow(Sender: TObject);
        Procedure FormClose(Sender: TObject; Var Action: TCloseAction);
        Procedure MMSaveFileClick(Sender: TObject);
        Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):
                          Boolean;
        Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);
    End;

Var
    CandidateSelectForm: TCandidateSelectForm;
    Vacancy: TVacancyInfo;

Implementation

{$R *.dfm}

Uses DateUtils, MainUnit;

Var
    FoundCandidatesHead: PCandidate = Nil;

Function IsCandidateSuitable(Candidate: TCandidateInfo): Boolean;
Var
    CandidateAge: Integer;
Begin
    CandidateAge := YearsBetween(Now, Candidate.BirthDate);
    IsCandidateSuitable := (AnsiUpperCase(String(Vacancy.Speciality)) =
                           AnsiUpperCase(String(Candidate.Speciality))) And
                           (AnsiUpperCase(String(Vacancy.Title)) =
                           AnsiUpperCase(String(Candidate.Title))) And
                           (Vacancy.IsHighEducationRequired And
                           Candidate.HasHighEducation Or Not
                           Vacancy.IsHighEducationRequired) And Not(Vacancy.MinAge >
                           CandidateAge) And Not(Vacancy.MaxAge < CandidateAge)
                           And Not(Vacancy.Salary < Candidate.Salary);
End;

Procedure SelectCandidates();
Var
    Temp: PCandidate;
Begin
```

```
    Temp := CandidateHead;
    While Temp <> Nil Do
    Begin
        If IsCandidateSuitable(Temp^.Info) Then
        Begin
            AddCandidate(Temp^.Info, FoundCandidatesHead);
            AddCandidateToListView(Temp^.Info, CandidateSelectForm.
                                ListViewCandidates);
        End;
        Temp := Temp.Next;
    End;
End;

Procedure TCandidateSelectForm.FormClose(Sender: TObject; Var Action: TCloseAction);
Begin
    ClearListView(ListViewVacancy);
    ClearListView(ListViewCandidates);
    DeleteCandidateList(FoundCandidatesHead);
End;

Function TCandidateSelectForm.FormHelp(Command: Word; Data: NativeInt;
                                    Var CallHelp: Boolean): Boolean;
Begin
    ShowInstruction();
    CallHelp := False;
End;

Procedure TCandidateSelectForm.FormKeyDown(Sender: TObject; Var Key: Word;
                                           Shift: TShiftState);
Begin
    If Key = VK_ESCAPE Then
        Close;
End;

Procedure TCandidateSelectForm.FormShow(Sender: TObject);
Begin
    AddVacancyToListView(Vacancy, ListViewVacancy);
    SelectCandidates();
    MMSaveFile.Enabled := ListViewCandidates.Items.Count > 0;
End;

Procedure TCandidateSelectForm.MMSaveFileClick(Sender: TObject);
Begin
    If SaveDialog.Execute Then
        SaveCandidateListToFile(FoundCandidatesHead, SaveDialog.FileName);
End;

End.
```

```
Unit DeficitUnit;

Interface

Uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls,
    Vcl.StdCtrls, Vcl.Menus;

Type
    TDeficitForm = Class(TForm)
        SaveDialog: TSaveDialog;
        ListView: TListView;
        MainMenu: TMainMenu;
        MMFile: TMenuItem;
        MMSaveFile: TMenuItem;
        LabelInfo: TLabel;
        Procedure FormShow(Sender: TObject);
        Procedure FormClose(Sender: TObject; Var Action: TCloseAction);
        Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);
        Procedure MMSaveFileClick(Sender: TObject);
        Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean):
                          Boolean;
    End;

Var
    DeficitForm: TDeficitForm;

Implementation

{$R *.dfm}

Uses CandidateListUnit, VacancyListUnit, MainUnit;

Type
    TTitle = Record
        Title: String[MAXLEN];
        VacancyCount, CandidateCount: Integer;
    End;

Var
    Head: PCandidate;
    TitleArr: Array Of TTitle;

Function FindArrPosition(Title: ShortString): Integer;
Var
    IsInArray: Boolean;
    I: Integer;
Begin
    IsInArray := False;
    I := 0;
    While Not IsInArray And (I < Length(TitleArr)) Do
    Begin
        If AnsiUpperCase(String(Title)) = AnsiUpperCase(String(TitleArr[I].Title))
        Then
            IsInArray := True;
        Inc(I);
    End;
    If IsInArray Then
        FindArrPosition := I - 1
```

```
    Else
        FindArrPosition := -1;
End;

Procedure FindAllVacancyTitles();
Var
    Temp: PVacancy;
    I: Integer;
Begin
    Temp := VacancyHead;
    While Temp <> Nil Do
    Begin
        I := FindArrPosition(Temp^.Info.Title);
        If I = -1 Then
        Begin
            SetLength(TitleArr, Length(TitleArr) + 1);
            TitleArr[High(TitleArr)].Title := Temp^.Info.Title;
            TitleArr[High(TitleArr)].VacancyCount := 1;
            TitleArr[High(TitleArr)].CandidateCount := 0;
        End
        Else
            Inc(TitleArr[I].VacancyCount);
        Temp := Temp.Next;
    End;
End;

Procedure CountCandidates();
Var
    Temp: PCandidate;
    I: Integer;
Begin
    Temp := CandidateHead;
    While Temp <> Nil Do
    Begin
        I := FindArrPosition(Temp^.Info.Title);
        If I <> -1 Then
            Inc(TitleArr[I].CandidateCount);
        Temp := Temp^.Next;
    End;
End;

Procedure FindDeficitCandidates();
Var
    I: Integer;
    Temp: PCandidate;
Begin
    For I := Low(TitleArr) To High(TitleArr) Do
        If TitleArr[I].VacancyCount > TitleArr[I].CandidateCount * 10 Then
        Begin
            Temp := CandidateHead;
            While Temp <> Nil Do
            Begin
                If AnsiUpperCase(String(Temp^.Info.Title)) =
                   AnsiUpperCase(String(TitleArr[I].Title)) Then
                Begin
                    AddCandidate(Temp^.Info, Head);
                    AddCandidateToListView(Temp^.Info, DeficitForm.ListView);
                End;
                Temp := Temp^.Next;
            End;
```

```
        End;
End;

Procedure TDeficitForm.FormClose(Sender: TObject; Var Action: TCloseAction);
Begin
    ClearListView(ListView);
    TitleArr := Nil;
    DeleteCandidateList(Head);
End;

Function TDeficitForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp:
                               Boolean): Boolean;
Begin
    ShowInstruction();
    CallHelp := False;
End;

Procedure TDeficitForm.FormKeyDown(Sender: TObject; Var Key: Word;
                                   Shift: TShiftState);
Begin
    If Key = VK_ESCAPE Then
        Close;
End;

Procedure TDeficitForm.FormShow(Sender: TObject);
Begin
    FindAllVacancyTitles();
    CountCandidates();
    FindDeficitCandidates();
    MMSaveFile.Enabled := Head <> Nil;
End;

Procedure TDeficitForm.MMSaveFileClick(Sender: TObject);
Begin
    If SaveDialog.Execute Then
        SaveCandidateListToFile(Head, SaveDialog.FileName);
End;

End.
```