Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ОТЧЕТ

по учебной практике

на тему:

БИРЖА ТРУДА

Студент:

гр. 251001 Лашкин В.Н.

Руководитель:

асс. Фадеева Е.Е.

Минск 2023

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc134988457)

[1. Задание на лабораторную работу 4](#_Toc134988458)

[1.1. Постановка задачи 4](#_Toc134988459)

[1.2. Эскиз ожидаемого результата 5](#_Toc134988460)

[2. Выполнение 7](#_Toc134988461)

[2.1. Разработка укрупненной схемы программы 7](#_Toc134988462)

[2.2. Разработка алгоритма чтения данных из файла 8](#_Toc134988463)

[2.3. Разработка алгоритма сортировки списка 9](#_Toc134988464)

[2.4. Разработка алгоритма поиска данных 10](#_Toc134988465)

[2.5. Разработка алгоритма добавления элемента в список 12](#_Toc134988466)

[2.6. Разработка алгоритма удаления элемента из списка 13](#_Toc134988467)

[2.7. Разработка алгоритма подбора вакансий для данного кандидата, вывода списка дефицитных вакансий 14](#_Toc134988468)

[2.8. Основные структуры данных 16](#_Toc134988469)

[2.9. Итоговый текст программы 19](#_Toc134988470)

# Задание на лабораторную работу

## Постановка задачи

Реализовать программу, работающую с типизированными файлами. При выборе пункта меню 1 должно производиться чтение данных из файлов, созданных и сохраненных ранее в соответствующие им по типу и назначению динамические списки. Дальнейшие манипуляции над данными выполняются в динамических списках. В момент, когда выбран пункт меню 10, перезаписываются типизированные файлы данными из соответствующих динамических списков.

Программа должна содержать пользовательское меню, состоящее из следующих пунктов:

1. Чтение данных из файла
2. Просмотр всего списка (выпадает подменю с именами списков для просмотра: списки вакансий и кандидатов)
3. Сортировка данных в соответствии с заданием
4. Поиск данных с использованием фильтров
5. Добавление данных в список (выпадает подменю с именами списков для добавления: списки вакансий и кандидатов)
6. Удаление данных из списка (выпадает подменю с именами списков для удаления: списки вакансий и кандидатов)
7. Редактирование данных (выпадает подменю с именами списков для редактирования: списки вакансий и кандидатов)
8. Подбор списка возможных вакансий для данного кандидата
9. Выход из программы без сохранения изменений
10. Выход с сохранением изменений

При реализации пункта 8 меню, полученные данные должны выводиться на экран и записываться в текстовый файл.

Имеется список фирм с вакансиями. Каждая запись списка содержит: название фирмы, наименование специальности, должность, оклад, количество дней отпуска, требования к нанимаемому: наличие высшего образования (да/нет), возрастной диапазон (min/max). Также имеется список кандидатов. Каждая запись списка содержит: ФИО кандидата, дату рождения, специальность, наличие высшего образования (да/нет), желаемую должность, min оклад. Требуется:

1. Для каждого кандидата подобрать список возможных вакансий;
2. Выдать список дефицитных вакансий (количество вакансий на должность ниже 10% процентов от количества кандидатов)

Предусмотреть возможность добавлять, удалять и корректировать записи из списков, а также просматривать списки полностью.

## Эскиз ожидаемого результата

После запуска программы перед пользователем появляется меню с выбором действия:

|  |
| --- |
| *Выберите пункт меню:*  *1. Чтение данных из файла*  *2. Просмотр всего списка*  *3. Сортировка данных в соответствии с заданием*  *4. Поиск данных с использованием фильтров*  *5. Добавление данных в список*  *6. Удаление данных из списка*  *7. Редактирование данных*  *8. Подбор списка возможных вакансий для данного кандидата*  *9. Выход из программы без сохранения изменений*  *10. Выход с сохранением изменений* |

Пользователь может выбирать пункты меню путем ввода чисел на клавиатуре. После выбора пункта *4. Поиск данных с использованием фильтров* появляется подменю:

|  |
| --- |
| *Выберите список, в который следует произвести поиск*  *1. Список вакансий*  *2. Список кандидатов* |

Пользователь может выбирать список, в котором будет произведен поиск. После выбора списка пользователю будет предложено выбрать поле записи, по которому будет произведен поиск:

|  |
| --- |
| *Выберите поле, по которому будет произведен поиск*  *1. Номер*  *2. Название фирмы*  *3. Специальность*  *4. Должность*  *5. Оклад*  *6. Количество дней отпуска*  *7. Наличие высшего образования*  *8. Возрастной диапазон* |

После выбора, например, поля *2. Название фирмы* пользователь может ввести значение этого поля и получить результат поиска:

|  |
| --- |
| Введите содержание поля  ГГОЛ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  | Номер| Фирма | Специальность | Должность | Оклад | Отпуск |Высшее образование| Возраст |  |\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 3| ГГОЛ| Профессор| Инспектор| 9000| 27| Нет| 25-50|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 4| ГГОЛ| Менеджер| Учитель| 12000| 27| Нет| 18-65|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 5| ГГОЛ| Тестировщик| Программист| 6000| 21| Нет| 18-50|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 28| ГГОЛ| Учитель| Профессор| 9000| 25| Да| 25-50|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 34| ГГОЛ| Программист| Программист| 3000| 28| Да| 30-60|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 44| ГГОЛ| Аналитик| Инженер| 6000| 23| Нет| 18-60|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 45| ГГОЛ| Менеджер| Программист| 9000| 28| Да| 30-60|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 47| ГГОЛ| Директор| Менеджер| 3000| 28| Да| 18-50|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|  | 52| ГГОЛ| Учитель| Директор| 4500| 27| Да| 30-50|  | \_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_| |

После завершения поиска пользователь возвращается в главное меню. Программа заканчивает свою работу, когда выбран пункт «*9. Выход из программы без сохранения изменений*» или «*10. Выход с сохранением изменений*».

# Выполнение

## Разработка укрупненной схемы программы

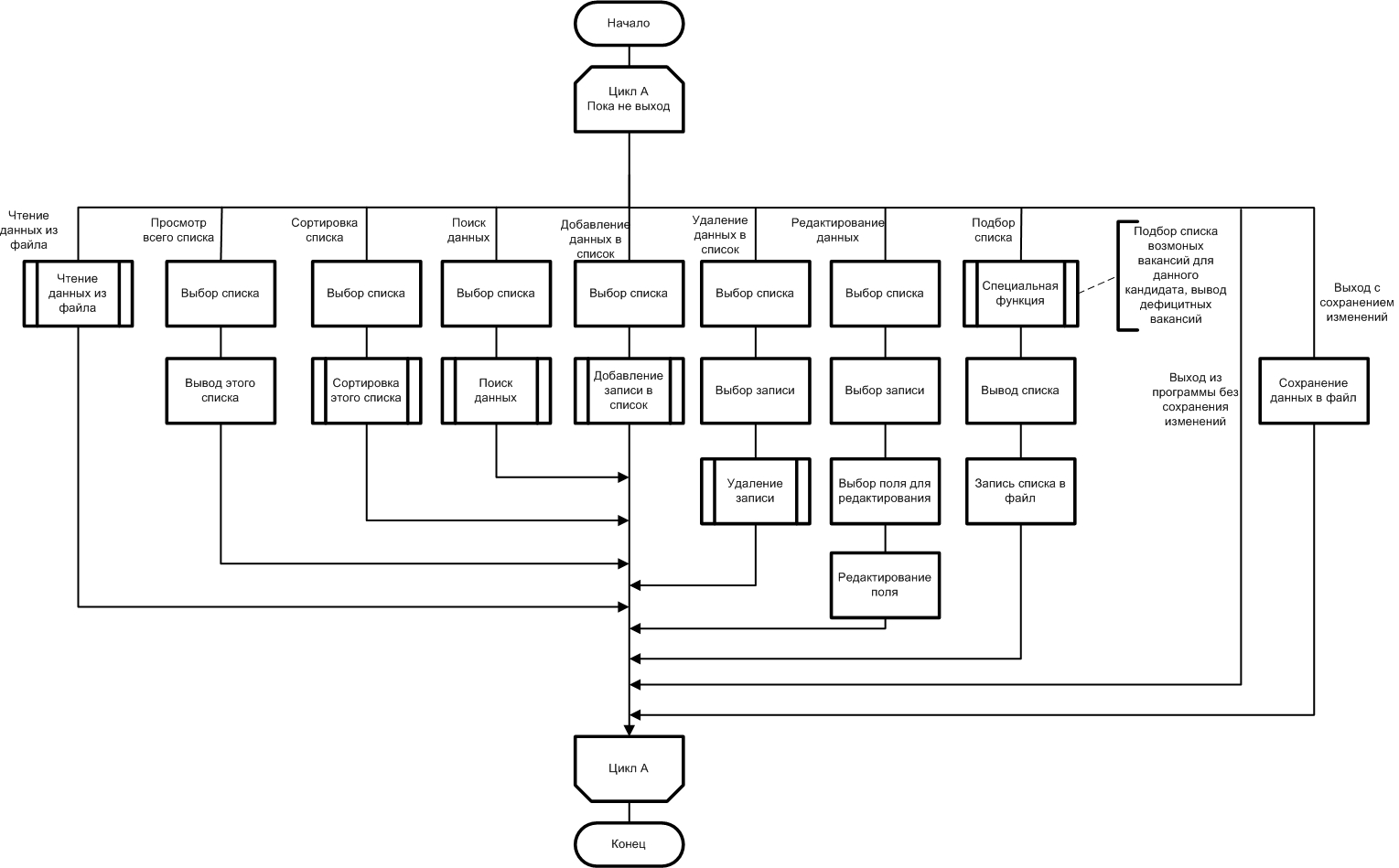


Рисунок 2.1 – Укрупненная схема программы

Эта схема отражает работу программы. После запуска программы пользователь может взаимодействовать с интерфейсом, который включает в себя меню из 10 пунктов. Программа завершается, когда пользователь выбирает «выход из программы».

## Разработка алгоритма чтения данных из файла



Рисунок 2.2 – Схема алгоритма чтения данных из файла

Эта схема отражает работу алгоритма чтения данных из файла. Пользователь вводит путь к файлу, который необходимо прочесть.

## Разработка алгоритма сортировки списка

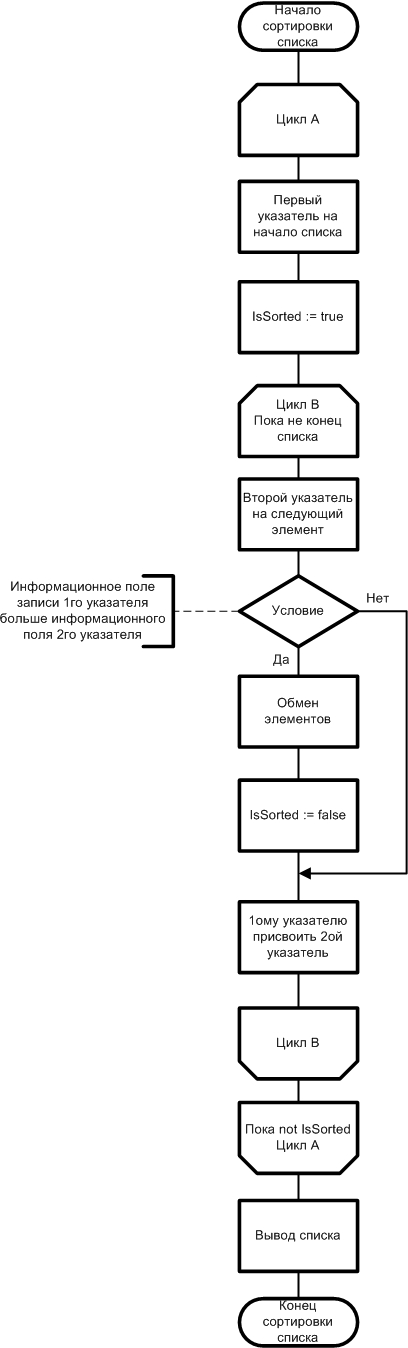


Рисунок 2.3 – Схема алгоритма сортировки списка

## Разработка алгоритма поиска данных

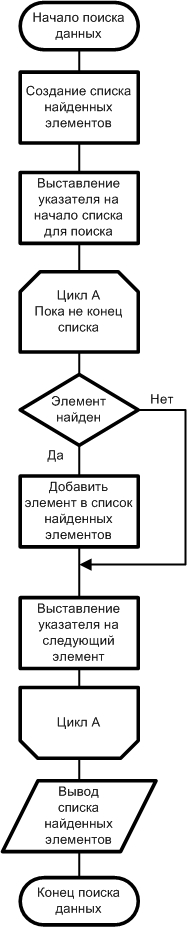


Рисунок 2.4 – Схема алгоритма поиска данных

Так как для некоторых полей найденных записей может быть несколько, то я создал однонаправленный список найденных записей. Устанавливается указатель на начало списка для поиска. Если элемент списка равен ключу, который ввел пользователь, то этот элемент заносится в список найденных элементов, выставляется указатель на следующий элемент. В конце цикла выставляется указатель на nil. Полученный список выводится на экран.

## Разработка алгоритма добавления элемента в список

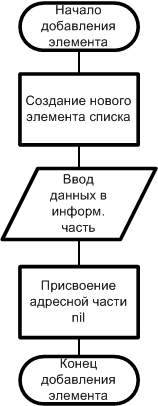


Рисунок 2.5 – Схема алгоритма добавления элемента в список

После того как пользователь выбрал список, в который он хочет добавить элемент, он вводит информационную часть записи добавляемого элемента.

## Разработка алгоритма удаления элемента из списка

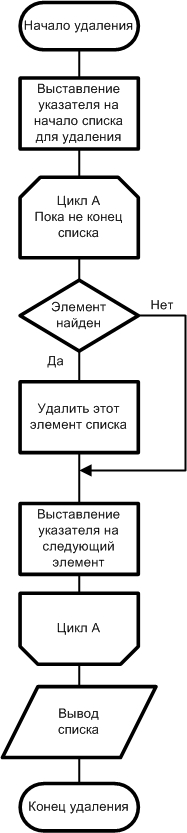


Рисунок 2.6 – Схема алгоритма удаления записи

Так как удаляемых элементов может быть несколько, то цикл А проходится по всему списку и удаляются только те записи, у которых информационная часть совпадает с введенным пользователем ключом.

## Разработка алгоритма подбора вакансий для данного кандидата, вывода списка дефицитных вакансий

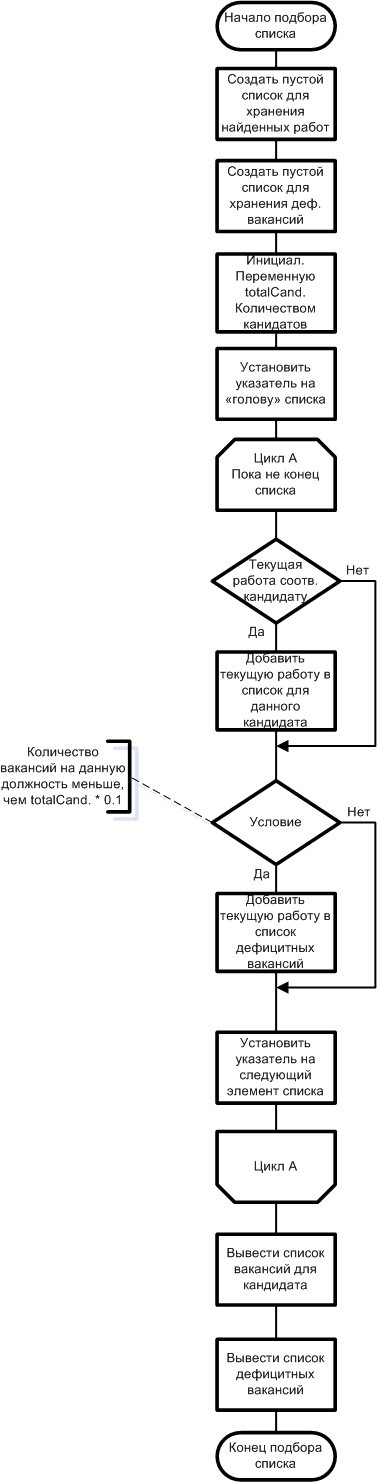


Рисунок 2.7 – Схема алгоритма подбора вакансий для данного кандидата, вывода списка дефицитных вакансий

Пользователь вводит должность, специальность, оклад. Создается список вакансий для данного кандидата, соответствующие введенным данным. Если число вакансий на данную должность меньше, чем 10% от общего количества кандидатов, то эта должность вносится в список дефицитных вакансий. В конце алгоритма эти два списка выводятся на экран.

## Основные структуры данных, процедур и функций

Таблица 2.1 – Описание основных структур данных

|  |  |  |
| --- | --- | --- |
| **Название** | **Назначение** | **Структура** |
| TRangeAge | Массив возрастного диапазона. Первый элемент массива – нижняя возрастная граница, второй элемент – верхняя | TRangeAge = array [min .. max] of Integer; |
| TVacancyInfo | Информация о вакансии | TVacancyInfo = record  Index: Integer;  Firm: String[20];  Specialization: String[20];  Post: String[20];  Salary: Integer;  Vacation: Integer;  HighEdu: boolean;  RangeAge: TRangeAge;  end; |
| TCandidateInfo | Информация о кандидате | TCandidateInfo = record  Index: Integer;  FIO: String[50];  BirthDay: TDateTime;  Specialization: String[20];  Post: String[20];  Salary: Integer;  HighEdu: boolean;  end; |
| TDeficitInfo | Информация о дефицитной должности. Post – должность, vCount – количество вакансий, где указана эта должность, cCount – количество кандидатов, указавших эту должность, Part – отношение количества вакансий к количеству кандидатов | TDeficitInfo = record  Post: String[20];  vCount: Integer;  cCount: Integer;  Part: Real;  end; |
| TVacancy | Список вакансий | TVacancy = record  Info: TVacancyInfo;  Adr: PVacancy;  end; |
| PVacancy | Указатель на список вакансий | PVacancy = ^TVacancy; |
| TCandidate | Список кандидатов | TCandidate = record  Info: TCandidateInfo;  Adr: PCandidate;  end; |
| PCandidate | Указатель на список кандидатов | PCandidate = ^TCandidate; |

Таблица 2.1 – Продолжение

|  |  |  |
| --- | --- | --- |
| TPossibleVacancy | Список возможных вакансий | TPossibleVacancy = record  Info: TVacancyInfo;  Adr: PPossibleVacancy;  end; |
| PPossibleVacancy | Указатель на список возможных вакансий | PPossibleVacancy = ^TPossibleVacancy; |
| TDeficit | Список дефицитных должностей | TDeficit = record  Info: TDeficitInfo;  Adr: PDeficit;  end; |
| PDeficit | Указатель на список дефицитных должностей | PDeficit = ^TDeficit; |

Для удобства основные типы данных были вынесены в модуль Data

Таблица 2.2 – Описание основных процедур и функций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| ClearConsole | Очищает консоль | procedure ClearConsole(const Position: Integer); | Position | Номер строки, с котором начать очищение |
| CorrectChoice | Корректный ввод выбора пункта меню | function CorrectChoice(const First, Last: Integer): Integer; | First | Первое допустимое число |
| Last | Последнее допустимое число |
| CorrectInteger | Корректный ввод числа | function CorrectInteger: Integer; |  |  |
| CorrectBirthDate | Корректный ввод даты рождения | function CorrectBirthDate: TDateTime; |  |  |
| YesNo | Переводит строку в тип Boolean | function YesNo: Boolean; |  |  |
| YesNoToStr | Переводит Boolean в строку | function YesNoToStr(var Input: Boolean): String; | Input | Булевская переменная |
| ReadFromFile | Читает данные из типизированных файлов и записывает их в списки | procedure ReadFromFile(vHead: PVacancy; cHead: PCandidate;  var vIndex, cIndex: Integer); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| vIndex | Максимальный номер в списке вакансий |
| cIndex | Максимальный номер в списке кандидатов |
| WriteInFile | Записывает данные из списков в типизированные файлы | procedure WriteInFile(vHead: PVacancy; cHead: PCandidate;  var vIndex, cIndex: Integer); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| vIndex | Максимальный номер в списке вакансий |
| cIndex | Максимальный номер в списке кандидатов |
| WriteInFileSpFunc | Записывает данные из списков для специальной функции в текстовый файл | procedure WriteInFileSpFunc(cTemp: PCandidate; vHead: PVacancy;  pvHead: PPossibleVacancy; dHead: PDeficit); | cTemp | Указатель на кандидата, для которого подбираем допустимые вакансии |
| vHead | «Голова» списка |
| pvHead | «Голова» списка |
| dHead | «Голова» списка |
| AddData | Добавляет записи в списки | procedure AddData(vHead: PVacancy; cHead: PCandidate;  var vIndex, cIndex: Integer); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| vIndex | Максимальный номер в списке вакансий |
| cIndex | Максимальный номер в списке кандидатов |

Таблица 2.2 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SearchData | Производит поиск записи по спискам с использованием фильтров | function SearchData(vHead: PVacancy; cHead: PCandidate;  var chTemp: Integer): Boolean; | vHead | «Голова» списка |
| cHead | «Голова» списка |
| chTemp | Список в котором происходит поиск, нужен для других подпрограмм |
| DeleteData | Удаляет записи из списков | procedure DeleteData(vHead: PVacancy; cHead: PCandidate); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| EditData | Редактирует записи из списков | procedure EditData(vHead: PVacancy; cHead: PCandidate); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| ShowData | Выводит записи из списков на экран | procedure ShowData(vHead: PVacancy; cHead: PCandidate); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| SortData | Сортирует записи в списках с использованием фильтров | procedure SortData(vHead: PVacancy; cHead: PCandidate); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| SpecialFuntion | Специальная функция. Выводит список возможных вакансий для данного кандидата и список дефицитных вакансий | procedure SpecialFuntion(vHead: PVacancy; cHead: PCandidate;  pvHead: PPossibleVacancy); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| pvHead | «Голова» списка |
| Initialize | Выделяет память под списки | procedure Initialize(var vHead: PVacancy; var cHead: PCandidate;  var pvHead: PPossibleVacancy; var dHead: PDeficit); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| pvHead | «Голова» списка |
| dHead | «Голова» списка |

Таблица 2.2 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Destroy | Освобождает память, выделенную под списки | procedure Destroy(vHead: PVacancy; cHead: PCandidate; pvHead: PPossibleVacancy;  dHead: PDeficit); | vHead | «Голова» списка |
| cHead | «Голова» списка |
| pvHead | «Голова» списка |
| dHead | «Голова» списка |
| Menu | Главное меню программы | procedure Menu; |  |  |

## Тестирование и отладка программы

Таблица 2.2. Прохождение тестов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Специфика тестирования | Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| Запуск программы | 1 | - | Появление главного меню | Тест пройден |
| Чтение данных из файла | 2 | - | Списки заполнятся данными из файлов | Тест пройден |
| Просмотр всего списка | 3 | Выберите список, который следует вывести  1. Список вакансий  2. Список кандидатов  1  Нажмите Enter... | Вывод на экран списка вакансий | Тест пройден |
| Сортировка данных в соответствии с заданием | 4 | Выбор списка кандидатов  Выберите поле, по которому будет произведен поиск  1. Номер  2. ФИО  3. Дата рождения  4. Специальность  5. Должность  6. Оклад  7. Наличие высшего образования  3  Нажмите Enter... | Вывод списка кандидатов, отсортированных по дате рождения | Тест пройден |
| Поиск данных с использованием фильтров | 5 | Выбор списка кандидатов  Выбор поля «Дата рождения»  Введите содержание поля  Введите год рождения  (1900-2023)  1994  Введите месяц рождения (1-12)  7  Введите день рождения  (1-31)  15 | Вывод результатов поиска | Тест пройден |

Таблица 2.2. – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Добавление данных в список (ручной способ) | 6 | Выбор списка вакансий  Введите название фирмы  БГУИР  Введите специализацию  Профессор истории  Введите должность  Профессор  Введите оклад  1500  Введите количество дней отпуска  28  Введите наличие высшего образования (да/нет)  да  Введите нижнюю границу возрастного диапазона  45  Введите верхнюю границу возрастного диапазона  75  Действие совершено успешно  Нажмите Enter... | Выбор списка вакансий,  ввод новой записи для добавления,  появление сообщения об успешном добавлении | Тест пройден |
| Добавление данных в список (автоматический способ) | 7 | Выбор списка вакансий  Введите количество  записей  20  Действие совершено успешно  Нажмите Enter... | Выбор списка вакансий,  ввод количества записей,  появление сообщения об успешном добавлении | Тест пройден |
| Удаление данных из списка | 8 | Выбор списка кандидатов  Выбор поля, по которому будет произведен поиск (ФИО)  Введите содержание поля  Иванов Иван Иванович  Введите индекс записи для удаления  1997  Действие совершено успешно  Нажмите Enter... | Выбор списка кандидатов,  выбор поля для поиска,  вывод результатов поиска,  ввод номера записи для удаления,  вывод сообщения об успешном удалении | Тест пройден |

Таблица 2.2. – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Редактирование данных | 9 | Выбор списка вакансий  Выбор поля, по которому будет произведен поиск (Название фирмы)  Введите содержание поля  ГГОЛ  Введите индекс записи для редактирования  387  Выбор поля, в котором будет произведено редактирование  7  Введите новое содержание поля  Введите нижнюю границу возрастного диапазона  18  Введите верхнюю границу возрастного диапазона  85  Действие совершено успешно  Нажмите Enter... | Выбор списка вакансий,  выбор поля для поиска,  вывод результатов поиска,  ввод номера записи для редактирования,  выбор поля, в котором будет произведено редактирование,  ввод нового содержания поля,  вывод сообщения об успешном редактировании | Тест пройден |
| Специальная функция | 10 | Выбор поля в списке кандидатов, по которому будет произведен поиск  7  Введите содержание поля  Да  Введите индекс записи для подбора списка возможных вакансий  121  Внести результат в текстовый файл (да/нет)  да  Действие совершено успешно  Нажмите Enter... | Выбор поля в списке кандидатов, по которому будет произведен поиск,  ввод содержания поля,  вывод на экран результатов поиска,  ввод номера записи для подбора списка возможных вакансий,  вывод на экран списка возможных вакансий для данного кандидата и списка дефицитных вакансий  согласие на запись результатов специальной функции в текстовый файл,  появление сообщения об успешной записи в файл | Тест пройден |

Таблица 2.2. – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выход из программы без сохранения изменений | 11 | Выберите пункт меню:  9  Нажмите Enter... | Типизированные файлы не перезаписываются,  память, выделенная под списки, освобождается,  программа завершает свою работу | Тест пройден |
| Выход с сохранением изменений | 12 | Выберите пункт меню:  10  Нажмите Enter... | Типизированные файлы перезаписываются,  память, выделенная под списки, освобождается,  программа завершает свою работу | Тест пройден |

## Руководство по установке и эксплуатации

Перед установкой необходимо убедиться, что компьютер соответствует минимальным системным требованиям:

1. ОС Windows 10
2. 64 Мб ОЗУ
3. 16 Мб свободного дискового пространства
4. Наличие клавиатуры

Установка программы, как и установка дополнительных средств, плагинов и специальных библиотек, не требуется, достаточно скопировать исполняемый файл LaborExchange.exe на жесткий диск компьютера. Запуск может происходить из любого места на жестком диске.

Для запуска программы, пользователь должен дважды щелкнуть левой кнопкой мыши по файлу с расширением LaborExchange.exe. После запуска программы на экране появится консольное окно, с которым пользователь может взаимодействовать при помощи клавиатуры:

|  |
| --- |
| *Выберите пункт меню:*  *1. Чтение данных из файла*  *2. Просмотр всего списка*  *3. Сортировка данных в соответствии с заданием*  *4. Поиск данных с использованием фильтров*  *5. Добавление данных в список*  *6. Удаление данных из списка*  *7. Редактирование данных*  *8. Подбор списка возможных вакансий для данного кандидата*  *9. Выход из программы без сохранения изменений*  *10. Выход с сохранением изменений* |

Ввод команд:

В консольном окне пользователь вводит номер команды при помощи клавиатуры, подтверждает ввод через нажатие клавиши Enter.

Обработка ошибок:

Если пользователь введет недопустимые значения, то выведется сообщение об ошибке.

Завершение работы:

Для выхода из программы пользователю необходимо ввести команду «*9. Выход из программы без сохранения изменений*» или «*10. Выход с сохранением изменений*» и нажмите клавишу Enter.

## Итоговый текст программы

**Содержание модуля Main**

program Main;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils, DateUtils, Windows,

Data in '..\units\Data.pas';

var

vHead: PVacancy;

cHead: PCandidate;

dHead, dTemp: PDeficit;

pvHead, pvTemp: PPossibleVacancy;

vIndex, cIndex, chMode: Integer;

procedure ClearConsole(const Position: Integer);

var

cursor: COORD;

r: cardinal;

begin

r := 10000;

cursor.X := 0;

cursor.Y := Position;

FillConsoleOutputCharacter(GetStdHandle(STD\_OUTPUT\_HANDLE), ' ', 80 \* r,

cursor, r);

cursor.X := 0;

cursor.Y := Position;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), cursor);

end;

function CorrectChoice(const First, Last: Integer): Integer;

var

Input: String;

Error: Integer;

Correct: Boolean;

begin

repeat

ReadLn(Input);

Val(Input, Result, Error);

Correct := Result in [First .. Last];

if (Error <> 0) or not Correct then

WriteLn(ErrorMessage);

until (Error = 0) and Correct;

end;

function CorrectInteger: Integer;

var

Input: String;

Error: Integer;

begin

repeat

ReadLn(Input);

Val(Input, Result, Error);

if Error <> 0 then

WriteLn(ErrorMessage);

until Error = 0;

end;

function CorrectBirthDate: TDateTime;

var

Year, Month, Day: Integer;

LeapYear: Boolean;

MaxDaysInMonth: Integer;

begin

repeat

WriteLn('Введите год рождения (1900-', YearOf(Now), ')');

Year := CorrectInteger;

if (Year < 1900) or (Year > YearOf(Now)) then

WriteLn(ErrorMessage);

until (Year >= 1900) and (Year <= YearOf(Now));

repeat

WriteLn('Введите месяц рождения (1-12)');

Month := CorrectInteger;

if (Month < 1) or (Month > 12) then

WriteLn(ErrorMessage);

until (Month >= 1) and (Month <= 12);

LeapYear := IsLeapYear(Year);

MaxDaysInMonth := MonthDays[LeapYear, Month];

repeat

WriteLn('Введите день рождения (1-', MaxDaysInMonth, ')');

Day := CorrectInteger;

if (Day < 1) or (Day > MaxDaysInMonth) then

WriteLn(ErrorMessage);

until (Day >= 1) and (Day <= MaxDaysInMonth);

Result := EncodeDate(Year, Month, Day);

end;

function YesNo: Boolean;

var

Input: String;

begin

Result := False;

repeat

ReadLn(Input);

Input := LowerCase(Trim(Input));

if Input = 'да' then

Result := True

else if Input = 'нет' then

Result := False

else

WriteLn(ErrorMessage);

until (Input = 'да') or (Input = 'нет');

end;

function YesNoToStr(var Input: Boolean): String;

begin

if Input then

Result := 'Да'

else

Result := 'Нет';

end;

procedure ReadFromFile(vHead: PVacancy; cHead: PCandidate;

var vIndex, cIndex: Integer);

var

FVacancy: FileVacancy;

FCandidate: FileCandidate;

vTemp: PVacancy;

vInfo: TVacancyInfo;

cTemp: PCandidate;

cInfo: TCandidateInfo;

begin

try

Assign(FVacancy, 'ListOfVacancies.lbe ');

Reset(FVacancy);

Read(FVacancy, vInfo);

vIndex := vInfo.Index;

vTemp := vHead;

while not EOF(FVacancy) do

begin

New(vTemp^.Adr);

Read(FVacancy, vInfo);

vTemp := vTemp^.Adr;

vTemp^.Info := vInfo;

vTemp^.Adr := Nil;

end;

Assign(FCandidate, 'ListOfCandidates.lbe ');

Reset(FCandidate);

Read(FCandidate, cInfo);

cIndex := cInfo.Index;

cTemp := cHead;

while not EOF(FCandidate) do

begin

New(cTemp^.Adr);

Read(FCandidate, cInfo);

cTemp := cTemp^.Adr;

cTemp^.Info := cInfo;

cTemp^.Adr := Nil;

end;

except

WriteLn('Файлы не найдены');

Sleep(1500);

end;

end;

procedure WriteInFile(vHead: PVacancy; cHead: PCandidate;

var vIndex, cIndex: Integer);

var

FVacancy: FileVacancy;

FCandidate: FileCandidate;

vTemp: PVacancy;

vInfo: TVacancyInfo;

cTemp: PCandidate;

cInfo: TCandidateInfo;

begin

Assign(FVacancy, 'ListOfVacancies.lbe ');

ReWrite(FVacancy);

vInfo.Index := vIndex;

Write(FVacancy, vInfo);

vTemp := vHead;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

Write(FVacancy, vTemp^.Info);

end;

CloseFile(FVacancy);

Assign(FCandidate, 'ListOfCandidates.lbe ');

ReWrite(FCandidate);

cInfo.Index := cIndex;

Write(FCandidate, cInfo);

cTemp := cHead;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

Write(FCandidate, cTemp^.Info);

end;

CloseFile(FCandidate);

end;

procedure WriteInFileSpFunc(cTemp: PCandidate; vHead: PVacancy;

pvHead: PPossibleVacancy; dHead: PDeficit);

var

FPossibleVacancy: TextFile;

vTemp: PVacancy;

pvTemp: PPossibleVacancy;

Temp: Boolean;

begin

Assign(FPossibleVacancy, 'ListOfPossibleCandidates.txt');

ReWrite(FPossibleVacancy);

WriteLn(FPossibleVacancy,

' \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кандидат\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ');

WriteLn(FPossibleVacancy, HeadCandidateList);

Write(FPossibleVacancy, HorizLineC);

with cTemp^.Info do

WriteLn(FPossibleVacancy, '|', Index:5, '|', FIO:32, '|',

DateToStr(BirthDay):13, '|', Specialization:19, '|', Post:13, '|',

Salary:9, '|', YesNoToStr(HighEdu):18, '|');

Write(FPossibleVacancy, HorizLineC);

WriteLn(FPossibleVacancy,

' \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Возможные вакансии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ');

Temp := False;

pvTemp := pvHead;

while pvTemp^.Adr <> nil do

begin

pvTemp := pvTemp^.Adr;

if not Temp then

WriteLn(FPossibleVacancy, HeadVacancyList);

Write(FPossibleVacancy, HorizLineV);

with pvTemp^.Info do

WriteLn(FPossibleVacancy, '|', Index:5, '|', Firm:15, '|',

Specialization:19, '|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-', RangeAge[max], '|');

Temp := True;

end;

Write(FPossibleVacancy, HorizLineV);

WriteLn(FPossibleVacancy,

' \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Дефицитные вакансии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ');

WriteLn(FPossibleVacancy, HeadVacancyList);

vTemp := vHead;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

dTemp := dHead;

while dTemp^.Adr <> nil do

begin

dTemp := dTemp^.Adr;

if (vTemp^.Info.Post = dTemp^.Info.Post) and (dTemp^.Info.Part < 0.1) then

begin

Write(FPossibleVacancy, HorizLineV);

with vTemp^.Info do

WriteLn(FPossibleVacancy, '|', Index:5, '|', Firm:15, '|',

Specialization:19, '|', Post:15, '|', Salary:11, '|', Vacation:12,

'|', YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

end;

end;

end;

WriteLn(FPossibleVacancy, HorizLineV);

CloseFile(FPossibleVacancy);

end;

procedure AddData(vHead: PVacancy; cHead: PCandidate;

var vIndex, cIndex: Integer);

var

Choice, Count, I: Integer;

vTemp: PVacancy;

cTemp: PCandidate;

begin

WriteLn(AddText);

Choice := CorrectChoice(1, 4);

vTemp := vHead;

cTemp := cHead;

case Choice of

1:

begin

while vTemp.Adr <> nil do

vTemp := vTemp^.Adr;

New(vTemp^.Adr);

vTemp := vTemp^.Adr;

with vTemp^.Info do

begin

WriteLn('Введите название фирмы');

ReadLn(Firm);

Trim(String(Firm));

WriteLn('Введите специализацию');

ReadLn(Specialization);

Trim(String(Specialization));

WriteLn('Введите должность');

ReadLn(Post);

Trim(String(Post));

WriteLn('Введите оклад');

Salary := CorrectInteger;

WriteLn('Введите количество дней отпуска');

Vacation := CorrectInteger;

WriteLn('Введите наличие высшего образования (да/нет)');

HighEdu := YesNo;

WriteLn('Введите нижнюю границу возрастного диапазона');

ReadLn(RangeAge[min]);

WriteLn('Введите верхнюю границу возрастного диапазона');

ReadLn(RangeAge[max]);

Index := vIndex;

Inc(vIndex);

end;

WriteLn(SuccessMessage);

vTemp^.Adr := nil;

end;

2:

begin

while cTemp.Adr <> nil do

cTemp := cTemp^.Adr;

New(cTemp^.Adr);

cTemp := cTemp^.Adr;

with cTemp^.Info do

begin

WriteLn('Введите ФИО');

ReadLn(FIO);

Trim(String(FIO));

BirthDay := CorrectBirthDate;

WriteLn('Введите специальность');

ReadLn(Specialization);

Trim(String(Specialization));

WriteLn('Введите должность');

ReadLn(Post);

Trim(String(Post));

WriteLn('Введите оклад');

Salary := CorrectInteger;

WriteLn('Введите наличие высшего образования (да/нет)');

HighEdu := YesNo;

Index := cIndex;

Inc(cIndex);

end;

WriteLn(SuccessMessage);

cTemp^.Adr := nil;

end;

3:

begin

WriteLn('Введите количество записей');

Count := CorrectInteger;

for I := 1 to Count do

begin

while vTemp.Adr <> nil do

vTemp := vTemp^.Adr;

New(vTemp^.Adr);

vTemp := vTemp^.Adr;

with vTemp^.Info do

begin

Firm := ShortString(TFirm[Random(7)]);

Specialization := ShortString(TSpecVacancy[Random(10)]);

Post := ShortString(TSpecVacancy[Random(10)]);

Salary := TSalary[Random(6)];

Vacation := TVacation[Random(5)];

HighEdu := Bool(Random(2));

RangeAge[min] := TAge[Random(3)];

RangeAge[max] := TAge[Random(3) + 3];

Index := vIndex;

Inc(vIndex);

end;

vTemp^.Adr := nil;

end;

WriteLn(SuccessMessage);

end;

4:

begin

WriteLn('Введите количество записей');

Count := CorrectInteger;

for I := 1 to Count do

begin

while cTemp.Adr <> nil do

cTemp := cTemp^.Adr;

New(cTemp^.Adr);

cTemp := cTemp^.Adr;

with cTemp^.Info do

begin

FIO := ShortString(TSurName[Random(5)] + ' ' + TName[Random(6)] +

' ' + TOtch[Random(6)]);

BirthDay := Random(22000) + 15000;

Specialization := ShortString(TSpecCandidate[Random(10)]);

Post := ShortString(TSpecCandidate[Random(10)]);

Salary := TSalary[Random(6)];

HighEdu := Bool(Random(2));

Index := cIndex;

Inc(cIndex);

end;

cTemp^.Adr := nil;

end;

WriteLn(SuccessMessage);

end;

end;

end;

function SearchData(vHead: PVacancy; cHead: PCandidate;

var chTemp: Integer): Boolean;

var

Choice, KeyInt: Integer;

KeyStr: String;

KeyDate: TDateTime;

KeyEdu, IsFound: Boolean;

KeyRange: TRangeAge;

vTemp: PVacancy;

cTemp: PCandidate;

begin

if chTemp = 0 then

begin

WriteLn(SearchText);

Choice := CorrectChoice(1, 2);

chTemp := Choice;

WriteLn(ContinueMessage);

ReadLn;

ClearConsole(0);

end

else

Choice := chTemp;

Result := True;

case Choice of

1:

begin

WriteLn(SearchMessageV);

Choice := CorrectChoice(1, 8);

WriteLn(ContinueMessage);

ReadLn;

ClearConsole(0);

IsFound := False;

WriteLn('Введите содержание поля');

vTemp := vHead;

case Choice of

1:

begin

KeyInt := CorrectInteger;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if Index = KeyInt then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

2:

begin

ReadLn(KeyStr);

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if String(Firm) = KeyStr then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

3:

begin

ReadLn(KeyStr);

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if String(Specialization) = KeyStr then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

4:

begin

ReadLn(KeyStr);

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if String(Post) = KeyStr then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

5:

begin

KeyInt := CorrectInteger;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if Salary = KeyInt then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

6:

begin

KeyInt := CorrectInteger;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if Vacation = KeyInt then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

7:

begin

KeyEdu := YesNo;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if HighEdu = KeyEdu then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

8:

begin

WriteLn('Введите нижнюю границу возрастного диапазона');

ReadLn(KeyRange[min]);

WriteLn('Введите верхнюю границу возрастного диапазона');

ReadLn(KeyRange[max]);

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

with vTemp^.Info do

if (KeyRange[min] = RangeAge[min]) and

(KeyRange[max] = RangeAge[max]) then

begin

if not IsFound then

WriteLn(HeadVacancyList);

Write(HorizLineV);

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19,

'|', Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineV);

end;

end;

end;

2:

begin

WriteLn(SearchMessageС);

Choice := CorrectChoice(1, 7);

WriteLn(ContinueMessage);

ReadLn;

ClearConsole(0);

IsFound := False;

WriteLn('Введите содержание поля');

cTemp := cHead;

case Choice of

1:

begin

KeyInt := CorrectInteger;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if Index = KeyInt then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

2:

begin

ReadLn(KeyStr);

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if String(FIO) = KeyStr then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

3:

begin

KeyDate := CorrectBirthDate;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if BirthDay = KeyDate then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

4:

begin

ReadLn(KeyStr);

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if String(Specialization) = KeyStr then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

5:

begin

ReadLn(KeyStr);

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if String(Post) = KeyStr then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

6:

begin

KeyInt := CorrectInteger;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if Salary = KeyInt then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

7:

begin

KeyEdu := YesNo;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

with cTemp^.Info do

if HighEdu = KeyEdu then

begin

if not IsFound then

WriteLn(HeadCandidateList);

Write(HorizLineC);

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay)

:13, '|', Specialization:19, '|', Post:13, '|', Salary:9,

'|', YesNoToStr(HighEdu):18, '|');

IsFound := True;

end;

end;

if not IsFound then

begin

WriteLn(NotFoundMessage);

Result := False;

end

else

Write(HorizLineC);

end;

end;

end;

end;

end;

procedure DeleteData(vHead: PVacancy; cHead: PCandidate);

var

Index, Choice, chTemp: Integer;

IsDeleted: Boolean;

vTempCurr, vTempPrev: PVacancy;

cTempCurr, cTempPrev: PCandidate;

begin

chTemp := 0;

if SearchData(vHead, cHead, chTemp) then

begin

Choice := chTemp;

WriteLn('Введите индекс записи для удаления');

Index := CorrectInteger;

IsDeleted := False;

case Choice of

1:

begin

vTempCurr := vHead;

vTempPrev := nil;

while (vTempCurr <> nil) and not IsDeleted do

begin

if (vTempCurr^.Info.Index = Index) and (vTempCurr <> vHead) then

begin

if vTempPrev = nil then

vHead := vTempCurr^.Adr

else

vTempPrev^.Adr := vTempCurr^.Adr;

Dispose(vTempCurr);

IsDeleted := True;

end;

if not IsDeleted then

begin

vTempPrev := vTempCurr;

vTempCurr := vTempCurr^.Adr;

end;

end;

if IsDeleted then

WriteLn(SuccessMessage)

else

WriteLn('Запись не найдена');

end;

2:

begin

cTempCurr := cHead;

cTempPrev := nil;

while (cTempCurr <> nil) and not IsDeleted do

begin

if (cTempCurr^.Info.Index = Index) and (cTempCurr <> cHead) then

begin

if cTempPrev = nil then

cHead := cTempCurr^.Adr

else

cTempPrev^.Adr := cTempCurr^.Adr;

Dispose(cTempCurr);

IsDeleted := True;

end;

if not IsDeleted then

begin

cTempPrev := cTempCurr;

cTempCurr := cTempCurr^.Adr;

end;

end;

if IsDeleted then

WriteLn(SuccessMessage)

else

WriteLn('Запись не найдена');

end;

end;

end;

end;

procedure EditData(vHead: PVacancy; cHead: PCandidate);

var

Choice, Index, KeyInt, chTemp: Integer;

IsFound, KeyEdu: Boolean;

KeyDate: TDateTime;

KeyRange: TRangeAge;

vTemp: PVacancy;

cTemp: PCandidate;

begin

chTemp := 0;

if SearchData(vHead, cHead, chTemp) then

begin

WriteLn('Введите индекс записи для редактирования');

Index := CorrectInteger;

IsFound := False;

vTemp := vHead;

cTemp := cHead;

Choice := chTemp;

case Choice of

1:

while (vTemp.Adr <> nil) and not IsFound do

begin

vTemp := vTemp^.Adr;

if vTemp^.Info.Index = Index then

IsFound := True;

end;

2:

while (cTemp.Adr <> nil) and not IsFound do

begin

cTemp := cTemp^.Adr;

if cTemp^.Info.Index = Index then

IsFound := True;

end;

end;

if IsFound then

begin

case Choice of

1:

begin

WriteLn(EditMessageV);

Choice := CorrectChoice(1, 7);

WriteLn('Введите новое содержание поля');

with vTemp^.Info do

case Choice of

1:

ReadLn(Firm);

2:

ReadLn(Specialization);

3:

ReadLn(Post);

4:

begin

KeyInt := CorrectInteger;

Salary := KeyInt;

end;

5:

begin

KeyInt := CorrectInteger;

Vacation := KeyInt;

end;

6:

begin

KeyEdu := YesNo;

HighEdu := KeyEdu;

end;

7:

begin

WriteLn('Введите нижнюю границу возрастного диапазона');

ReadLn(KeyRange[min]);

WriteLn('Введите верхнюю границу возрастного диапазона');

ReadLn(KeyRange[max]);

RangeAge[min] := KeyRange[min];

RangeAge[max] := KeyRange[max];

end;

end;

WriteLn(SuccessMessage);

end;

2:

begin

WriteLn(EditMessageС);

Choice := CorrectChoice(1, 6);

WriteLn('Введите новое содержание поля');

with cTemp^.Info do

case Choice of

1:

ReadLn(FIO);

2:

begin

KeyDate := CorrectBirthDate;

BirthDay := KeyDate;

end;

3:

ReadLn(Specialization);

4:

ReadLn(Post);

5:

begin

KeyInt := CorrectInteger;

Salary := KeyInt;

end;

6:

begin

KeyEdu := YesNo;

HighEdu := KeyEdu;

end;

end;

WriteLn(SuccessMessage);

end;

end;

end

else

WriteLn(NotFoundMessage);

end;

end;

procedure ShowData(vHead: PVacancy; cHead: PCandidate);

var

Choice: Integer;

vTemp: PVacancy;

cTemp: PCandidate;

begin

WriteLn(ShowText);

Choice := CorrectChoice(1, 2);

WriteLn(ContinueMessage);

ReadLn;

ClearConsole(0);

case Choice of

1:

begin

WriteLn(HeadVacancyList);

vTemp := vHead;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

Write(HorizLineV);

with vTemp^.Info do

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19, '|',

Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

end;

WriteLn(HorizLineV);

end;

2:

begin

WriteLn(HeadCandidateList);

cTemp := cHead;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

Write(HorizLineC);

with cTemp^.Info do

WriteLn('|', Index:5, '|', FIO:32, '|', DateToStr(BirthDay):13, '|',

Specialization:19, '|', Post:13, '|', Salary:9, '|',

YesNoToStr(HighEdu):18, '|');

end;

WriteLn(HorizLineC);

end;

end;

end;

procedure SortData(vHead: PVacancy; cHead: PCandidate);

var

Choice: Integer;

vCurr, vNext, vTemp: PVacancy;

cCurr, cNext, cTemp: PCandidate;

begin

WriteLn(SortText);

Choice := CorrectChoice(1, 2);

WriteLn(ContinueMessage);

ReadLn;

ClearConsole(0);

case Choice of

1:

begin

WriteLn(SortMessageV);

Choice := CorrectChoice(1, 8);

case Choice of

1:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.Index < vCurr^.Info.Index) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

2:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.Firm < vCurr^.Info.Firm) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

3:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.Specialization <

vCurr^.Info.Specialization) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

4:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.Post < vCurr^.Info.Post) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

5:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.Salary < vCurr^.Info.Salary) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

6:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.Vacation < vCurr^.Info.Vacation) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

7:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.HighEdu < vCurr^.Info.HighEdu) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

8:

begin

vCurr := vHead^.Adr;

vHead^.Adr := nil;

while vCurr <> nil do

begin

vNext := vCurr^.Adr;

vTemp := vHead;

while (vTemp^.Adr <> nil) and

(vTemp^.Adr^.Info.RangeAge[min] <

vCurr^.Info.RangeAge[min]) do

vTemp := vTemp^.Adr;

vCurr^.Adr := vTemp^.Adr;

vTemp^.Adr := vCurr;

vCurr := vNext;

end;

end;

end;

WriteLn(SuccessMessage);

end;

2:

begin

WriteLn(SortMessageС);

Choice := CorrectChoice(1, 7);

case Choice of

1:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.Index < cCurr^.Info.Index) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

2:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.FIO < cCurr^.Info.FIO) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

3:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.BirthDay < cCurr^.Info.BirthDay) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

4:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.Specialization <

cCurr^.Info.Specialization) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

5:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.Post < cCurr^.Info.Post) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

6:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.Salary < cCurr^.Info.Salary) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

7:

begin

cCurr := cHead^.Adr;

cHead^.Adr := nil;

while cCurr <> nil do

begin

cNext := cCurr^.Adr;

cTemp := cHead;

while (cTemp^.Adr <> nil) and

(cTemp^.Adr^.Info.HighEdu < cCurr^.Info.HighEdu) do

cTemp := cTemp^.Adr;

cCurr^.Adr := cTemp^.Adr;

cTemp^.Adr := cCurr;

cCurr := cNext;

end;

end;

end;

WriteLn(SuccessMessage);

end;

end;

end;

procedure SpecialFuntion(vHead: PVacancy; cHead: PCandidate;

pvHead: PPossibleVacancy);

var

vTemp: PVacancy;

cTemp, cTempFound: PCandidate;

pvTemp: PPossibleVacancy;

dTemp: PDeficit;

KeyInt, chMode: Integer;

IsFoundI, IsFoundV, IsFoundD, Temp: Boolean;

begin

chMode := 2;

if SearchData(vHead, cHead, chMode) then

begin

WriteLn('Введите индекс записи для подбора списка возможных вакансий');

KeyInt := CorrectInteger;

cTemp := cHead;

IsFoundI := False;

IsFoundV := False;

while (cTemp^.Adr <> nil) and not IsFoundI do

begin

cTemp := cTemp^.Adr;

if cTemp^.Info.Index = KeyInt then

begin

vTemp := vHead;

pvTemp := pvHead;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

if (vTemp^.Info.Specialization = cTemp^.Info.Specialization) and

(vTemp^.Info.Post = cTemp^.Info.Post) and

(vTemp^.Info.Salary >= cTemp^.Info.Salary) and

(vTemp^.Info.HighEdu = cTemp^.Info.HighEdu) and

((vTemp^.Info.RangeAge[min] <= YearsBetween(cTemp^.Info.BirthDay,

Now)) and (vTemp^.Info.RangeAge[max] >=

YearsBetween(cTemp^.Info.BirthDay, Now))) then

begin

New(pvTemp^.Adr);

pvTemp := pvTemp^.Adr;

pvTemp.Info := vTemp.Info;

pvTemp^.Adr := nil;

IsFoundV := True;

end;

end;

IsFoundI := True;

end;

end;

cTempFound := cTemp;

if IsFoundV and IsFoundI then

begin

WriteLn('Возможные вакансии для данного кандидата: ');

pvTemp := pvHead;

Temp := False;

while pvTemp^.Adr <> nil do

begin

pvTemp := pvTemp^.Adr;

if not Temp then

WriteLn(HeadVacancyList);

Write(HorizLineV);

with pvTemp^.Info do

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19, '|',

Post:15, '|', Salary:11, '|', Vacation:12, '|', YesNoToStr(HighEdu)

:18, '|', RangeAge[min]:10, '-', RangeAge[max], '|');

Temp := True;

end;

Write(HorizLineV);

end

else

begin

WriteLn(NotFoundMessage);

end;

end;

vTemp := vHead;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

dTemp := dHead;

IsFoundI := False;

while (dTemp^.Adr <> nil) and not IsFoundI do

begin

dTemp := dTemp^.Adr;

if vTemp^.Info.Post = dTemp^.Info.Post then

begin

Inc(dTemp^.Info.vCount);

IsFoundI := True;

end;

end;

if not IsFoundI then

begin

New(dTemp^.Adr);

dTemp := dTemp^.Adr;

dTemp^.Info.Post := vTemp^.Info.Post;

dTemp^.Info.vCount := 1;

dTemp^.Info.cCount := 0;

dTemp^.Adr := nil;

end;

end;

cTemp := cHead;

while cTemp^.Adr <> nil do

begin

cTemp := cTemp^.Adr;

dTemp := dHead;

IsFoundI := False;

while (dTemp^.Adr <> nil) and not IsFoundI do

begin

dTemp := dTemp^.Adr;

if cTemp^.Info.Post = dTemp^.Info.Post then

begin

Inc(dTemp^.Info.cCount);

IsFoundI := True;

end;

end;

if not IsFoundI then

begin

New(dTemp^.Adr);

dTemp := dTemp^.Adr;

dTemp^.Info.Post := cTemp^.Info.Post;

dTemp^.Info.vCount := 0;

dTemp^.Info.cCount := 1;

dTemp^.Adr := nil;

end;

end;

IsFoundD := False;

dTemp := dHead;

while dTemp.Adr <> nil do

begin

dTemp := dTemp^.Adr;

with dTemp^.Info do

try

Part := vCount / cCount;

// WriteLn('[ ', vCount, ' , ', cCount, ' ]');

if Part < 0.1 then

IsFoundD := True;

except

Part := MaxInt;

end;

end;

if IsFoundD then

begin

WriteLn('Список дефицитных вакансий:');

WriteLn(HeadVacancyList);

vTemp := vHead;

while vTemp^.Adr <> nil do

begin

vTemp := vTemp^.Adr;

dTemp := dHead;

while dTemp^.Adr <> nil do

begin

dTemp := dTemp^.Adr;

if (vTemp^.Info.Post = dTemp^.Info.Post) and (dTemp^.Info.Part < 0.1)

then

begin

Write(HorizLineV);

with vTemp^.Info do

WriteLn('|', Index:5, '|', Firm:15, '|', Specialization:19, '|',

Post:15, '|', Salary:11, '|', Vacation:12, '|',

YesNoToStr(HighEdu):18, '|', RangeAge[min]:10, '-',

RangeAge[max], '|');

end;

end;

end;

WriteLn(HorizLineV);

end

else

WriteLn('Дефицитные вакансии не найдены');

if IsFoundV or IsFoundD then

begin

WriteLn('Внести результат в текстовый файл (да/нет)');

if YesNo then

begin

WriteLn(SuccessMessage);

WriteInFileSpFunc(cTempFound, vHead, pvHead, dHead);

end;

end;

end;

procedure Initialize(var vHead: PVacancy; var cHead: PCandidate;

var pvHead: PPossibleVacancy; var dHead: PDeficit);

begin

New(vHead);

vHead^.Adr := nil;

New(cHead);

cHead^.Adr := nil;

New(pvHead);

pvHead^.Adr := nil;

New(dHead);

dHead^.Adr := nil;

end;

procedure Destroy(vHead: PVacancy; cHead: PCandidate; pvHead: PPossibleVacancy;

dHead: PDeficit);

var

vTemp: PVacancy;

cTemp: PCandidate;

pvTemp: PPossibleVacancy;

dTemp: PDeficit;

begin

vTemp := vHead^.Adr;

while vTemp <> nil do

begin

vHead^.Adr := vTemp^.Adr;

Dispose(vTemp);

vTemp := vHead^.Adr;

end;

vHead^.Adr := nil;

Dispose(vHead);

cTemp := cHead^.Adr;

while cTemp <> nil do

begin

cHead^.Adr := cTemp^.Adr;

Dispose(cTemp);

cTemp := cHead^.Adr;

end;

cHead^.Adr := nil;

Dispose(cHead);

pvTemp := pvHead^.Adr;

while pvTemp <> nil do

begin

pvHead^.Adr := pvTemp^.Adr;

Dispose(pvTemp);

pvTemp := pvHead^.Adr;

end;

pvHead^.Adr := nil;

Dispose(pvHead);

dTemp := dHead^.Adr;

while dTemp <> nil do

begin

dHead^.Adr := dTemp^.Adr;

Dispose(dTemp);

dTemp := dHead^.Adr;

end;

dHead^.Adr := nil;

Dispose(dHead);

end;

procedure Menu;

var

Choice: Integer;

begin

repeat

WriteLn(HelloMessage);

Choice := CorrectChoice(1, 10);

WriteLn(ContinueMessage);

ReadLn;

ClearConsole(0);

case Choice of

1:

ReadFromFile(vHead, cHead, vIndex, cIndex);

2:

begin

ShowData(vHead, cHead);

WriteLn(ContinueMessage);

ReadLn;

end;

3:

begin

SortData(vHead, cHead);

WriteLn(ContinueMessage);

ReadLn;

end;

4:

begin

chMode := 0;

SearchData(vHead, cHead, chMode);

WriteLn(ContinueMessage);

ReadLn;

end;

5:

begin

AddData(vHead, cHead, vIndex, cIndex);

WriteLn(ContinueMessage);

ReadLn;

end;

6:

begin

DeleteData(vHead, cHead);

WriteLn(ContinueMessage);

ReadLn;

end;

7:

begin

EditData(vHead, cHead);

WriteLn(ContinueMessage);

ReadLn;

end;

8:

begin

pvTemp := pvHead^.Adr;

while pvTemp <> nil do

begin

pvHead^.Adr := pvTemp^.Adr;

Dispose(pvTemp);

pvTemp := pvHead^.Adr;

end;

dTemp := dHead^.Adr;

while dTemp <> nil do

begin

dHead^.Adr := dTemp^.Adr;

Dispose(dTemp);

dTemp := dHead^.Adr;

end;

pvHead^.Adr := nil;

SpecialFuntion(vHead, cHead, pvHead);

WriteLn(ContinueMessage);

ReadLn;

end;

9:

;

10:

WriteInFile(vHead, cHead, vIndex, cIndex);

end;

ClearConsole(0);

until (Choice = 9) or (Choice = 10);

end;

begin

Randomize;

vIndex := 1;

cIndex := 1;

Initialize(vHead, cHead, pvHead, dHead);

Menu;

Destroy(vHead, cHead, pvHead, dHead);

end.

**Содержание модуля Data**

unit Data;

interface

const

min = 1;

max = 2;

ErrorMessage = 'Некорректный ввод';

SuccessMessage = 'Действие совершено успешно';

ContinueMessage = 'Нажмите Enter...';

NotFoundMessage = 'Поиск не дал результатов';

HelloMessage = 'Выберите пункт меню:' + #10#13 + '1. Чтение данных из файла' +

#10#13 + '2. Просмотр всего списка' + #10#13 +

'3. Сортировка данных в соответствии с заданием' + #10#13 +

'4. Поиск данных с использованием фильтров' + #10#13 +

'5. Добавление данных в список' + #10#13 + '6. Удаление данных из списка' +

#10#13 + '7. Редактирование данных' + #10#13 +

'8. Подбор списка возможных вакансий для данного кандидата' + #10#13 +

'9. Выход из программы без сохранения изменений' + #10#13 +

'10. Выход с сохранением изменений';

SearchMessageV = 'Выберите поле, по которому будет произведен поиск' + #10#13

+ '1. Номер' + #10#13 + '2. Название фирмы' + #10#13 + '3. Специальность' +

#10#13 + '4. Должность' + #10#13 + '5. Оклад' + #10#13 +

'6. Количество дней отпуска' + #10#13 + '7. Наличие высшего образования' +

#10#13 + '8. Возрастной диапазон';

SearchMessageС = 'Выберите поле, по которому будет произведен поиск' + #10#13

+ '1. Номер' + #10#13 + '2. ФИО' + #10#13 + '3. Дата рождения' + #10#13 +

'4. Специальность' + #10#13 + '5. Должность' + #10#13 + '6. Оклад' + #10#13

+ '7. Наличие высшего образования';

EditMessageV = 'Выберите поле, в котором будет произведено редактирование' +

#10#13 + '1. Название фирмы' + #10#13 + '2. Специальность' + #10#13 +

'3. Должность' + #10#13 + '4. Оклад' + #10#13 + '5. Количество дней отпуска'

+ #10#13 + '6. Наличие высшего образования' + #10#13 +

'7. Возрастной диапазон';

EditMessageС = 'Выберите поле, в котором будет произведено редактирование' +

#10#13 + '1. ФИО' + #10#13 + '2. Дата рождения' + #10#13 +

'3. Специальность' + #10#13 + '4. Должность' + #10#13 + '5. Оклад' + #10#13

+ '6. Наличие высшего образования';

SortMessageV = 'Выберите поле, в котором будет произведена сортировка' +

#10#13 + '1. Номер' + #10#13 + '2. Название фирмы' + #10#13 +

'3. Специальность' + #10#13 + '4. Должность' + #10#13 + '5. Оклад' + #10#13

+ '6. Количество дней отпуска' + #10#13 + '7. Наличие высшего образования' +

#10#13 + '8. Возрастной диапазон';

SortMessageС = 'Выберите поле, в котором будет произведена сортировка' +

#10#13 + '1. Номер' + #10#13 + '2. ФИО' + #10#13 + '3. Дата рождения' +

#10#13 + '4. Специальность' + #10#13 + '5. Должность' + #10#13 + '6. Оклад'

+ #10#13 + '7. Наличие высшего образования';

AddText = 'Выберите список, в который следует добавить данные' + #10#13 +

'1. Список вакансий' + #10#13 + '2. Список кандидатов' + #10#13 + '3. Список вакансий (автоматический ввод)' + #10#13 + '4. Список кандидатов (автоматический ввод)';

ShowText = 'Выберите список, который следует вывести' + #10#13 +

'1. Список вакансий' + #10#13 + '2. Список кандидатов';

DeleteText = 'Выберите список, в котором следует произвести удаление' + #10#13

+ '1. Список вакансий' + #10#13 + '2. Список кандидатов';

SortText = 'Выберите список, в котором следует произвести сортировку' + #10#13

+ '1. Список вакансий' + #10#13 + '2. Список кандидатов';

SearchText = 'Выберите список, в который следует произвести поиск' + #10#13 +

'1. Список вакансий' + #10#13 + '2. Список кандидатов';

HeadHorizLine =

' \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ '

+ #10#13;

HeadVacancyList = HeadHorizLine + '|' + 'Номер' + '|' + ' Фирма ' +

'|' + ' Специальность ' + '|' + ' Должность ' + '|' + ' Оклад '

+ '|' + ' Отпуск ' + '|' + 'Высшее образование' + '|' +

' Возраст ' + '|';

HeadCandidateList = HeadHorizLine + '|' + 'Номер' + '|' +

' ФИО ' + '|' + 'Дата рождения' + '|' +

' Специальность ' + '|' + ' Должность ' + '|' + ' Оклад ' + '|' +

'Высшее образование' + '|';

HorizLineV =

'|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|'

+ #10#13;

HorizLineC =

'|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|'

+ #10#13;

TSpecVacancy: array[0 .. 9] of String = ('Программист', 'Маркетолог', 'Инженер', 'Директор', 'Профессор', 'Учитель', 'Тестировщик', 'Аналитик', 'Инспектор', 'Менеджер');

TSpecCandidate: array[0 .. 9] of String = ('Программист', 'Маркетолог', 'Программист', 'Директор', 'Программист', 'Программист', 'Программист', 'Программист', 'Инспектор', 'Программист');

TName: array[0 .. 5] of String = ('Иван', 'Сергей', 'Евгений', 'Павел', 'Михал', 'Владимир');

TOtch: array[0 .. 5] of String = ('Палыч', 'Сергеевич', 'Константинович', 'Владимирович', 'Николаевич', 'Дмитриевич');

TSurName: array[0 .. 4] of String = ('Новик', 'Дроздов', 'Иванов', 'Осипов', 'Высоцкий');

TSalary: array[0 .. 5] of Integer = (3000, 4500, 6000, 9000, 12000, 15000);

TFirm: array[0 .. 6] of String = ('ГГОЛ', 'БГУИР', 'SoftTeco', 'EPAM', 'Andersen', 'HardTeco', 'Itransition');

TVacation: array[0 .. 4] of Integer = (21, 23, 25, 27, 28);

TAge: array[0 .. 5] of Integer = (18, 25, 30, 50, 60, 65);

type

TRangeAge = array [min .. max] of Integer;

TVacancyInfo = record

Index: Integer;

Firm: String[20];

Specialization: String[20];

Post: String[20];

Salary: Integer;

Vacation: Integer;

HighEdu: boolean;

RangeAge: TRangeAge;

end;

TCandidateInfo = record

Index: Integer;

FIO: String[50];

BirthDay: TDateTime;

Specialization: String[20];

Post: String[20];

Salary: Integer;

HighEdu: boolean;

end;

TDeficitInfo = record

Post: String[20];

vCount: Integer;

cCount: Integer;

Part: Real;

end;

PVacancy = ^TVacancy;

TVacancy = record

Info: TVacancyInfo;

Adr: PVacancy;

end;

PCandidate = ^TCandidate;

TCandidate = record

Info: TCandidateInfo;

Adr: PCandidate;

end;

PPossibleVacancy = ^TPossibleVacancy;

TPossibleVacancy = record

Info: TVacancyInfo;

Adr: PPossibleVacancy;

end;

PDeficit = ^TDeficit;

TDeficit = record

Info: TDeficitInfo;

Adr: PDeficit;

end;

FileVacancy = file of TVacancyInfo;

FileCandidate = file of TCandidateInfo;

FilePossibleVacancy = file of TVacancyInfo;

implementation

end.