Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

# оТЧЕТ

к практической работе

на тему:

Учебная практика

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил  Студент гр. 351001 |  | Радецкий А.Н. | |
| Проверил |  |  | |
|  |  | Фадеева Е. Е. |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2024

**СОДЕРЖАНИЕ**

**1.ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ 3**

1.1. Теоретические сведения по теме лабораторной работы 3

1.2. Постановка задачи 4

1.3. Эскиз ожидаемого результата……………………………………………………………………….4

**2. ВЫПОЛЕНИЕ 7**

2.1 Алгоритм работы программы7

2.2 Алгоритм чтения файлов7

2.3 Алгоритм отображения списков7

2.4 Алгоритм добавления в список7

2.5 Алгоритм удаления элемента из списка8

2.6 Алгоритм редактирования списка8

2.7 Алгоритм сортировки списка8

2.8 Алгоритм поиска элемента в списке 9

2.9 Алгоритм работы специальной функции 9

2.10 Алгоритм сохранения в файл9

2.11 Определение основных подпрограмм и их описание 10

2.12 Выделение основных структур данных 12

2.13 Тестирование и отладка программы13

2.14 Руководство по эксплуатации15

2.15 Код программы 21

**3.** **ПРИЛОЖЕНИЯ57**

ПРИЛОЖЕНИЕ А57

ПРИЛОЖЕНИЕ Б67

1. **ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

**1.1 Теоретические сведения по теме лабораторной работы**

**Связный список** — базовая динамическая структура данных, состоящая из [узлов](https://ru.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B5%D0%BB_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), содержащих [данные](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [ссылки](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед [массивом](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

**Физический файл ­­­­­­­–** поименованная область памяти на внешнем носителе, в которой хранится некоторая информация.

**Логический файл** – представление физического файла в программе.

**Файловый тип –** произвольная последовательность элементов, длина которая заранее неизвестна, а конкретизируется в процессе выполнения программы.

**Способы доступа:**

* Прямой
* Последовательный

**Виды переменных файлового типа:**

* Текстовые файлы
* Типизированные файлы
* Нетипизированные файлы

**Типизированный файл** — это файл, в котором записаны идентичные структуры. Типизированный файл можно создать только программным путем. Он состоит из записей фиксированной длины. Содержимое файла — это машинное представление данных. Типизированные файлы так же называют двоичными т.к. они содержат информацию в двоичном коде. С каждым типизированным файлом связана переменная - указатель. Указатель определяет смещение от начала файла в количестве прочитанных или записанных записей. При открытии файла значение указателя равно нулю, т.е. не выполнено ни одной операции ввода/вывода. Если только одна запись записана или прочитана, то значение указателя равно единице.

**Текстовый файл ­–** текстовые файлы обеспечивают простой и удобный способ сохранения текстовых данных. Они обеспечивают простой механизм для чтения и записи численных данных, сохраненных как текст.

**1.2. Постановка задачи**

Мной было разработано ТЗ, представленное в **Приложение А.**

* 1. **Эскиз ожидаемого результата**

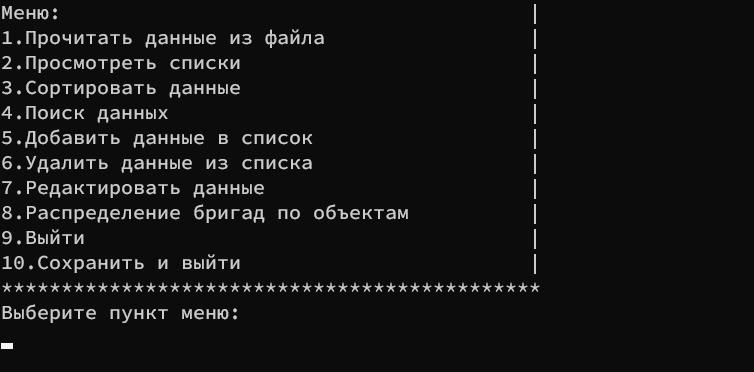
Вид пользовательского меню должен выглядеть так:

Рисунок 1.1 – Пользовательское меню.

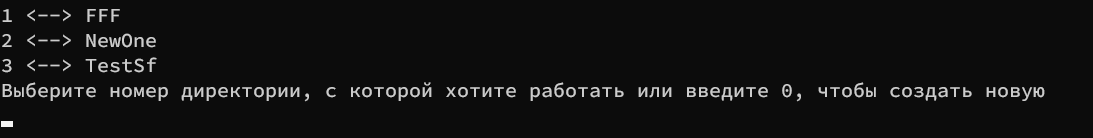
После нажатия выбора файла для чтения должно быть отображено подменю выбора файла.

Рисунок 1.2 – Подменю выбора файла

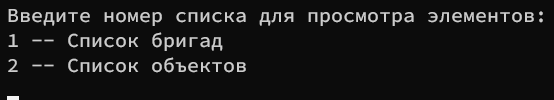
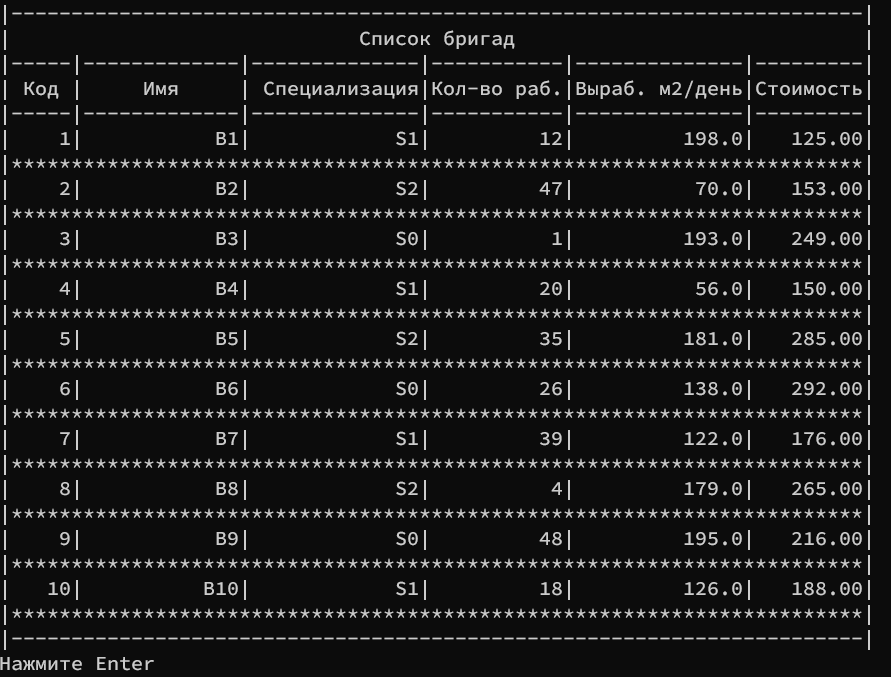
При выборе пунктов меню, соответствующее действие которых может быть применено к различным объектам (различным спискам) будет выведено подменю выбора соответствующего объекта (списка).

Рисунок 1.3 – Подменю выбора списка для просмотра.

Вывод информации об объектах списка должно быть отображено следующим образом:

Рисунок 1.4 – Просмотр информации списка

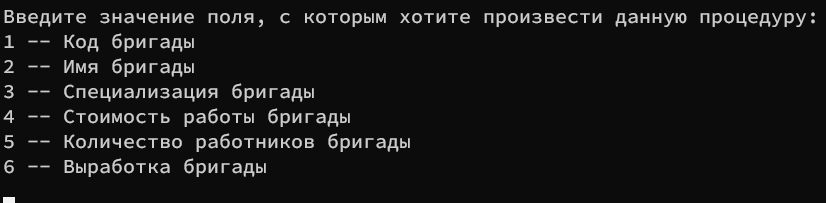
Выбор способа взаимодействия со списком через значение определенного поля элемента должно быть отображено следующим образом:

Рисунок 1.6 – Подменю выбора параметра элемента списка для поиска

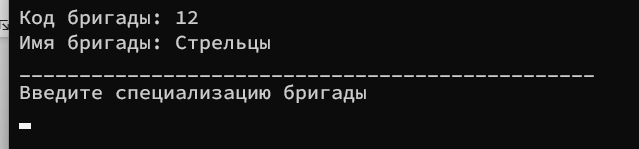
При выборе пункта добавления данных в список будет выведено подменю выбора списка для добавления. Затем информация, которая будет вводится в качестве данных элементов списка будет постепенно выведена на экран.

Рисунок 1.7 – Вывод информации при добавлении нового элемента

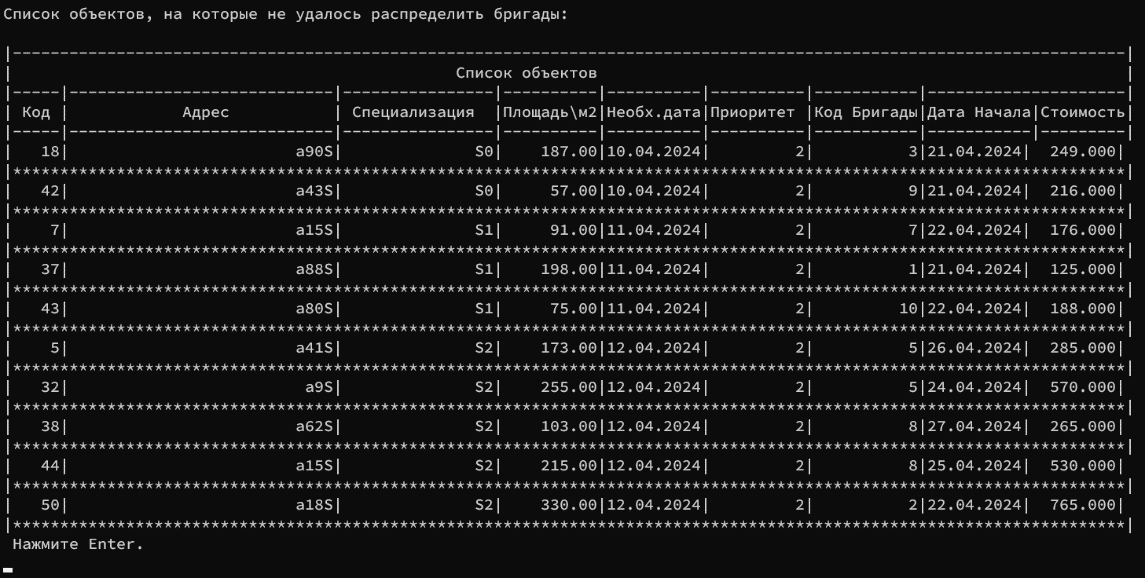
При выборе пункта распределения бригад по объектам будет выведен результат работы. В случае неудачи будут выведены объекты, на которые не получится распределить бригады, иначе будет выведено сообщение об успехе.

Рисунок 1.8 – Вывод информации при активации спец-функции

**2. ВЫПОЛЕНИЕ.**

**Разработка алгоритма по ГОСТ 19.701-90**

Схемы разработанных алгоритмов представлены в **Приложении 2.**

**2.1 Алгоритм работы программы**

Работа программы построена на основе работы отдельных вспомогательных алгоритмов в зависимости от выбора пункта меню пользователем. После отображения основного меню программа ожидает ввод пользователем номера действия, которое в последствии будет выполнено.

Данный алгоритм представлен в **Приложении 2, схема Б.1, схема Б2**

**2.2 Алгоритм чтения файлов**

Выполнение алгоритма выполнения разделено на 2 этапа. Первый этап представляет собой поэлементное чтение файла, второй – создание списка на основе прочитанных данных. В случае уже прочитанного файла повторное чтение запрещено.

Данный алгоритм представлен в **Приложении 2, схема Б.3**

**2.3 Алгоритм отображения списков**

В зависимости от выбора пользователем списка для отображения будет выведена вся информация об элементах, находящимся в данном списке. Для этого в цикле, начиная с первого элемента в списке, будет перемещен указатель на последующие элементы, при этом выводя информацию, об элементе, на который ссылается данный указатель.

Данный алгоритм представлен в **Приложении 2, схема Б.4, схема Б.5**

**2.4 Алгоритм добавления в список**

При выполнении алгоритма добавления в список сначала создается вспомогательный элемент, значения которого формируются на основании введенных пользователем данных. Затем если пользователь подтвердил добавление данного элемента в список, то на следующий за “головой” списка элемент перемещается вспомогательный указатель, после чего под следующий за головой элемент аллоцируется новый участок памяти, значения которого заполняются в соответствии со значениями вспомогательного элемента. Далее указателю на элемент, следующему за новым элементом присваивается значение вспомогательного указателя. Память, выделенная под вспомогательный элемент, освобождается.

Данный алгоритм представлен в **Приложении 2, схема Б.6**

**2.5 Алгоритм удаления элемента из списка**

При выполнении алгоритма удаления из списка изначально программа ожидает осуществляет поиск элемента с введенным пользователем кодом. В случае наличия данного элемента, будет ожидаться подтверждение удаления со стороны пользователя, после чего в случае подтверждения указатель предшествующего элемента будет переставлен на элемент, следующий за удаляемым, а выделенная память будет освобождена.

Данный алгоритм представлен в **Приложении 2, схема Б.8**

**2.6 Алгоритм редактирования списка**

Алгоритм редактирования списка основан на изменении соответствующей информации в зависимости от ввода пользователя. Изначально производится поиск элемента, который необходимо отредактировать. Затем создается вспомогательный элемент, значения которого заполняет пользователь. После ввода значений программа ожидает подтверждение редактирования данного элемента. Если редактирование было подтверждено, то данные вспомогательного элемента переносятся в редактируемый элемент. Память, выделенная под вспомогательный элемент, освобождается.

Данный алгоритм представлен в **Приложении 2, схема Б.9**

**2.7 Алгоритм сортировки списка**

Алгоритм сортировки списка основан на алгоритме сортировке простым выбором.

Метод сортировки прямым выбором основан на следующих правилах:

* Выбирается элемент с наименьшим ключом.
* Он меняется местами с первым элементом.
* Затем эти операции повторяются с оставшимися 𝑛−1 элементами, 𝑛−2 элементами и так далее, до тех пор, пока не останется один, самый большой элемент.

Для случая сортировки по убыванию сравниваются элементы с наибольшим ключом.

Данный алгоритм представлен в **Приложении 2, схема Б.10**

**2.8 Алгоритм поиска элемента в списке**

Алгоритм основан на пошаговом прохождении всего списка в цикле, при этом выводя все элементы, которые соответствуют введенным пользователем значениям.

Данный алгоритм представлен в **Приложении 2, схема Б.7**

**2.9 Алгоритм работы специальной функции**

Для начала работы специальной функции необходимо наличие хотя бы по одному элементу в каждом списке. Изначально инициализируются массивы с информацией из списков. После происходит сортировка массивов по специализации. Затем в цикле для каждого множества объектов заданной специализации будут найдены все бригады с данной специализацией, после чего начнется распределение бригад по объектам. Алгоритм распределения бригад по объектам является алгоритмом, основанным на принципе динамического программирования, где оптимальным решением будет распределение бригад по объектам как можно раньше к заданной дате, при этом за наименьшее количество времени. Алгоритм в цикле обрабатывает массив с датами начала работы на каждом объекте, после чего на каждом шаге обновляет его в соответствии с предыдущими значениями для нахождения оптимума.

Данный алгоритм представлен в **Приложении 2, схема Б.11**

**2.10 Алгоритм сохранения в файл**

Алгоритм основан на прохождении списков в цикле и записи значений их элементов в типизированный файл, выбор которого зависит от выбора пользователя.

Данный алгоритм представлен в **Приложении 2, схема Б.12**

Все представленные алгоритмы проходят строгую валидацию на корректность введенных значений.

**2.11 Определение основных подпрограмм и их описание**

Таблица 2.1 ­– Основные подпрограммы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя подпрограммы | Назначение подпрограммы | Заголовок подпрограммы | Имя параметра | Назначение параметра | |
| InitLists | Инициализация списков | Procedure InitLists(var brigadeHead: PtBrig;  var constrHead, constrTail: PtConstr); | brigadeHead | Указатель на голову списка бригад | |
| constrHead | Указатель на голову списка объектов | |
| constrTail | Указатель на хвост списка объектов | |
| InputInfo | Ввод информации для элементов | procedure InputInfo(const typeDate: TListData; var newElem: TTempNode; var fStopInput: boolean); | typeDate | Переменная, определяющая тип списка. | |
| newElem | Буферный элемент для предварительной записи данных | |
| fStopInput | Флаг прекращения работы | |
| AddElemToBrigList | Добавление элемента в список бригад | Procedure AddElemToBrigList(var brigadeHead: PtBrig); | brigadeHead | Указатель на голову списка бригад | |
| AddElemToConstrList | Добавление элемента в список объектов | procedure AddElemToConstrList(const constrHead: PtConstr;  var constrTail: PtConstr); | constrHead | Указатель на голову списка объектов | |
| constrTail | Указатель на хвост списка объектов | |
| SearchBrigElem | Поиск элементов в списке бригад | procedure SearchBrigElem(const brigHead: PtBrig); | brigHead | Указатель на голову списка бригад | |
|  | |  |
| SearchConstrElуm | Поиск элементов в списке объектов | procedure SearchConstrElem(const constrHead: PtConstr); | constrHead | Указатель на голову списка объектов | |
| SearchInit | Инициализация поиска | procedure SearchInit(const data: TListData; var fields: TFields; var Types: TTypes); | data | Тип данных элемента для поиска | |
| fields | Тип идентифицирующий пользовательский выбор поля для поиска | |
| Types | Тип элемента для поиска | |

Таблица 2.1 ­– Основные подпрограммы (Продолжение)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Search | Поиск | Procedure Search (const brigHead: PtBrig; const constrHead: PtConstr); | brigHead | Указатель на голову списка бригад |
| constrHead | Указатель на голову списка объектов |
| DeleteBrigElem | Удаление элемента из списка бригад | procedure DeleteBrigElem(var brigHead: PtBrig; const constrHead: PtConstr); | brigHead | Указатель на голову списка бригад |
| constrHead | Указатель на голову списка объектов |
| DeleteConstrElem | Удаление элемента из списка объектов | procedure DeleteConstrElem(var constrHead: PtConstr); | constrHead | Указатель на голову списка объектов |
| InitDel | Инициализация удаления | procedure InitDel(var brigHead: PtBrig; var constrHead: PtConstr; var constrTail: PtConstr); | brigHead | Указатель на голову списка бригад |
| constrHead | Указатель на голову списка объектов |
| constrTail | Указатель на хвост списка объектов |
| Sort | Инициализация сортировки | procedure Sort(var brigHead: PtBrig; var constrHead: PtConstr); | brigHead | Указатель на голову списка бригад |
| constrHead | Указатель на голову списка объектов |
| EditBrigList | Изменение элемента списка бригад | procedure EditBrigList(const brigHead: PtBrig); | brigHead | Указатель на голову списка бригад |
| EditConstrlist | Изменение элемента списка объектов | procedure EditConstrList(const constrHead: PtConstr); | constrHead | Указатель на голову списка объектов |
| InitEditLists | Инициализация изменения списков | procedure InitEditLists(const brigHead: PtBrig; const constrHead: PtConstr); | brigHead | Указатель на голову списка бригад |
| constrHead | Указатель на голову списка объектов |
| InitSortElemBrig | Инициализация сортировки списка бригад | procedure InitSortElemBrig(var tempElem: PtComp; const fields: TFields; const brigElem: PtBrig); | tempElem | Значения сравниваемого элемента |
| fields | Вид поля, по которому идет сравнение |
| brigElem | Значение элемента списка для сравнения |
| InitSortElemConstr | Инициализация сортировки списка объектов | procedure InitSortElemConstr(var tempElem: PtComp; const fields: TFields; var constrElem: PtConstr); | tempElem | Значения сравниваемого элемента |
| fields | Вид поля, по которому идет сравнение |
| constrElem | Значение элемента списка для сравнения |
| SpecialFunc | Распределение бригад по объектам | procedure SpecialFunc(const constrHead: ptConstr; const brigHead: ptBrig); | brigHead | Указатель на голову списка бригад |
| constrHead | Указатель на голову списка объектов |
| MainLogicSpec | Основная логика распределения бригад | procedure MainLogicSpec(var ConstrArr: TConstrArr; var BrigArr: TBrigArr); | ConstrArr | Массив объектов |
| BrigArr | Массив бригад |
| ReadInConstrList | Чтение в список объектов | Procedure ReadInConstrList(var constrHead, constrTail: PtConstr; const FConstr: TCFile); | constrHead | Указатель на голову списка объектов |
| constrTail | Указатель на хвост списка объектов |
| FConstr | Типизированный файл |

**2.12 Выделение основных структур данных**

Таблица 2.2 – Основные структуры данных

|  |  |  |
| --- | --- | --- |
| Имя идентификатора структуры | Назначение структуры | Тип структуры |
| TString | Строка для ввода значений | string[20] |
| TConstrInf | Запись для хранения полей элементов списка объектов | TConstrInf = record  constrCode: Integer;  constrAddr: TString;  constrSpec: TString;  constrArea: Real;  constrStDateToDo: TdateTime;  constrPriority: String [1];  constrBrigadCode: Integer;  constrStDate: TdateTime;  constrPrice: Currency;  end; |
| TBrigInf | Запись для хранения полей элементов списка бригад | TBrigInf = record  brigCode: Integer;  brigName: TString;  brigSpec: TString;  brigMemCnt: Integer;  brigSqDev: Real;  brigPrice: Currency;  end; |
| TTempNode | Запись для временного хранения данных списков | TBrigadeNode = record  brigInf: TBrigInf;  next: PtBrig;  end; |
| TConstrNode | Запись для хранения полей элементов списка объектов и поля, ссылающегося на след элемент | TConstrNode = record  constrInf: TConstrInf;  next: PtConstr;  end; |
| TBrigadeNode | Запись для хранения полей элементов списка бригад и поля, ссылающегося на след элемент | TBrigadeNode = record  brigInf: TBrigInf;  next: PtBrig;  end; |

**2.13 Тестирование и отладка программы**

Таблица 2.3 ­– Тестирование и отладка программы

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| 1.Запуск программы | – | Отображение основного меню в соответствии с рисунком 1.1 | Отображение меню в соответствии с эскизом программы.  Тест пройден успешно. |
| 2. Чтение из файлов | Выбор соответствующего файла для чтения | Отображение сохраненной информации в программе | Перенос сохраненных ранее данных в списки.  Тест пройден успешно. |
| 3.Попытка повторного чтения из файла | Выбор пункта меню чтение из файла | Отображение сообщение о невозможном повторном чтении файла. | Отображено сообщение, что файл прочитан:  «Данные файлов уже прочитаны»  Тест пройден успешно. |
| 4.Добавление в список | Ввод информации элемента списка | Появление данного элемента в списке. | Элемент добавлен в список для последующей работы.  Тест пройден успешно. |
| 5.Добавление в список элемента с некорректно введенными данными | Ввод некорректной информации | Отображение сообщения о некорректности введенных данных | Отображено сообщение о некорректности введенных данных:  «Данные введены неверно повторите попытку»  Тест пройден успешно. |
| 5.Просмотр списка | Выбор пункта меню просмотра списка | Отображение информации элементов списка в соответствии с рисунком 1.4 | Отображена информация элементов списка в соответствии с эскизом.  Тест пройден успешно**.** |
| 6.Поиск существующего элемента в списке | Ввод параметра элемента для поиска | Вывод соответствующего параметру элемента списка | Отображена информация найденного элемента.  Тест пройден успешно |
| 7.Поиск несуществующего элемента в списке | Ввод параметра элемента для поиска | Вывод соответствующего сообщения об отсутствии элемента в списке | Выведено сообщение:  «Элемент не найден»  Тест пройден успешно |
| 8.Сортировка списка бригад по возрастанию количества членов бригады | Ввод соответствующих значений пунктов подменю для начала сортировки | Отсортированный список согласно с параметрами | Список отсортирован корректно в соответствии с условием теста.  Тест пройден успешно |
| 9.Сортировка пустого списка | Ввод соответствующих значений пунктов подменю для начала сортировки | Прекращение работы сортировки из-за отсутствия элементов в списке | Сортировка на пустом списке не была проведена.  Тест пройден успешно |
| 10. Редактирование данных списка | Ввод соответствующих данных для редактирования | Изменение отображаемой информация у элемента | Ифнормация данного элемента была редактирована.  Тест пройден успешно |
| 11.Редактирование информации (код назначенной бригады) на несуществующую. | Ввод соответствующих данных для редактирования | Вывод сообщения о невозможности изменить данную информацию на введенную | Выведено сообщенне о невозможности отредактировать данную информацию.  Тест пройден успешно |
| 12.Удаление существующего элемента из списка | Ввод кода существующего элемента | Удаление элемента из списка | Элемента из списка удален  Тест пройден успешно |
| 13. Удаление несуществующего элемента из списка | Ввод кода несуществующего элемента | Вывод сообщения о невозможности удалить элемент по причине его отсутствия | Выведено сообщение о невозможности удалить данный элемент  «Данный элемент невозможно удалить»  Тест пройден успешно |
| 14. Активация специальной функции для непустых списков | Выбор соответствующего пункта меню при наличии элементов в списках. | Вывод результатов специальной функции | Были выведены результаты специальной функции.  Тест пройден успешно. |
| 15. Активация специальной функции при наличии пустого списка | Выбор соответствующего пункта меню при наличии пустого списках | Вывод о невозможности распределить объекты по бригадам по причине отсутствия элементов в одном из списков | Отображение сообщения о невозможности активации спеиальной функции: «Не удалось активировать спецфункцию, так как один из списков пуст» |
| 16.Сохранение результатов спец функции в файл | Предшествующая активации спец функции, выбор файла для сохранения | Отображение информации в выбранном файле | Информация о результатах работы спецфункции была отображена.  Тест пройден успешно. |
| 17.Проверка условия спец функции при выводе о невозможности распределить бригады на объекты при условии разницы фактической даты начала работы и необходимой на более чем 10 дней | Ввод информации в списки:  Бригад:(Производительность равна 12 м2/день)  Объектов (площадь работ 12342,14 м2; факт. дата 12,13) | Вывод о неудачном распределении бригады по объектам | Было отображено сообщение о неудачном распределении бригады по объектам.  Тест пройден успешно. |
| 18.Выход из программы без сохранения | Выбор соответствующего пункта меню | Завершение работы | Программа была завершена  Тест пройден успешно. |
| 19.Завершение работы при выходе из программы с сохранением | Выбор соответствующего пункта меню | Завершение работы с предварительным сохранением результата | Программа была завершена  Тест пройден успешно. |
| 20. Сохранение результатов сессии при выходе из программы с сохранением | Выбор соответствующего пункта меню, файла для сохранения | Сохранение информации в папке с файлами, содержащими информацию из сохраненных сессиях | В папку была сохранена информация о новой сессии. |

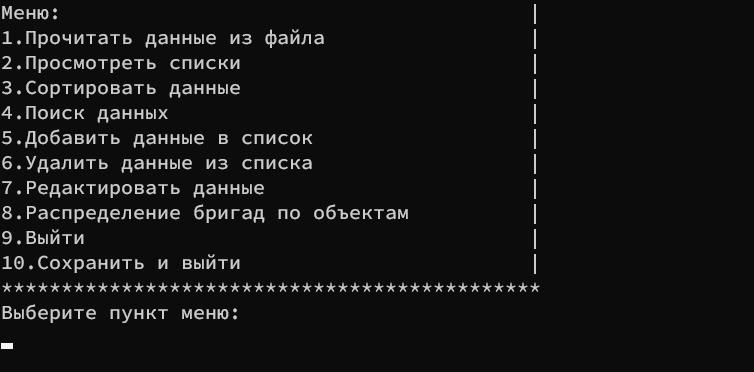
**2.14 Руководство по эксплуатации**

Для установки данной программы ваш компьютер должен соответствовать требованиям:

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

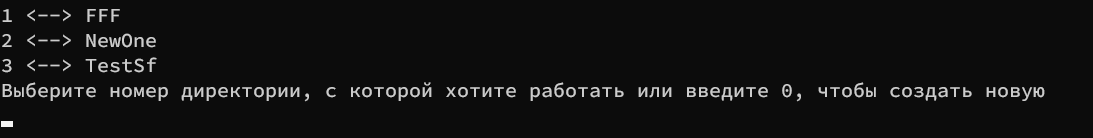
Чтобы установить программу, нужно скачать exe-файл Construction Manager.exe на жесткий диск компьютера. Для начала работы с программой нужно дважды щелкнуть по файлу Construction Manager.exe, после чего приложение будет запущено.

Взаимодействие пользователя с данной программой происходит через клавиатуру. На экране будут отображены инструкции к возможному выполнению определенных действий.

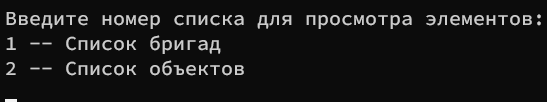
После запуска программы пользователя будет встречать начальное меню, состоящее из 10 пунктов.

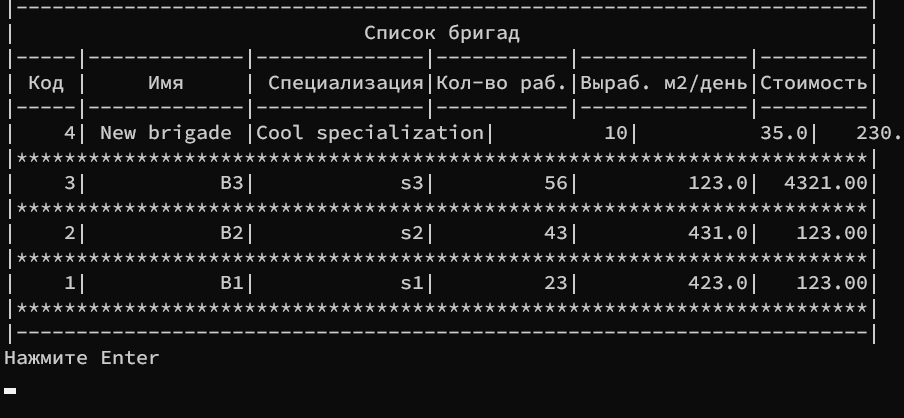
Для выбора определенного пункта меню необходимо ввести его номер. Все действия выбора подтверждаются нажатием кнопки Enter.

**Чтение из файла:**

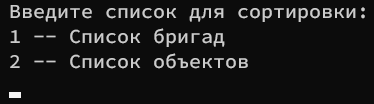
 Для работы с сохраненными результатами необходимо выбрать соответствующий пункт меню “Прочитать данные из файла”, после чего будет выведено подменю выбора файла для прочтения.

**Просмотр списков:**

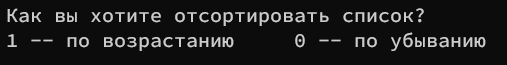
Для просмотра списков необходимо выбрать соответствующий пункт меню “Просмотреть списки”, после чего будет выведено подменю выбора списка для просмотра информации.

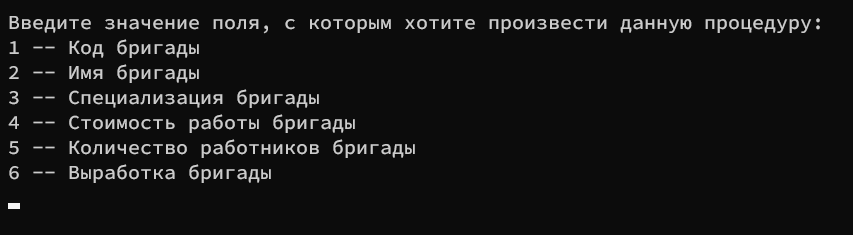
После выбора соответствующего списка будет отображена вся информация об элементах данного списка.

**Сортировка данных:**

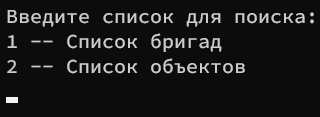
Для сортировки данных необходимо выбрать соответствующий пункт меню “Сортировать данные”, после чего будет выведено подменю выбора списка для сортировки.

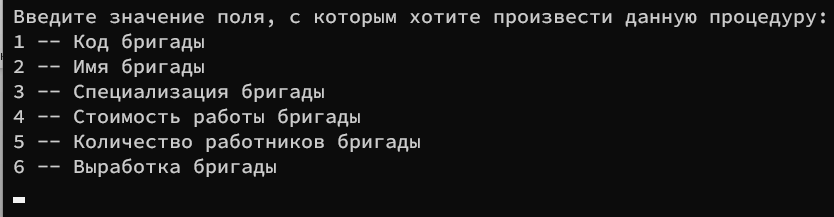
Далее пользователь может выбрать способ сортировки (по возрастанию/по убыванию)



После будет предложено выбрать вид поля с информацией для сортировки.

**Поиск данных:**

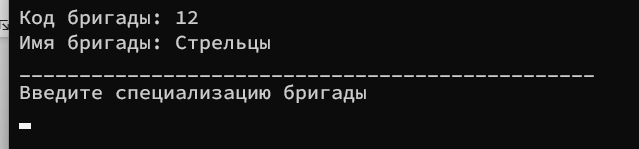
****Для поиска данных необходимо выбрать соответствующий пункт меню “Поиск данных”, после чего будет выведено подменю выбора списка для поиска.

После выбора списка будет отображено подменю выбора значения поля, по которому пользователь желает произвести поиск.

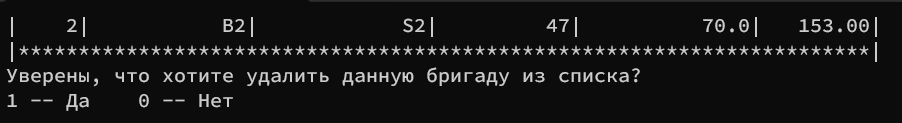
Затем необходимо ввести значение поля для поиска, после чего будут выведены результаты поиска.

**Добавление элементов в список**

При выборе пункта добавления данных в список будет выведено подменю выбора списка для добавления. Затем информация, которая будет вводится в качестве данных элементов списка будет постепенно выведена на экран.

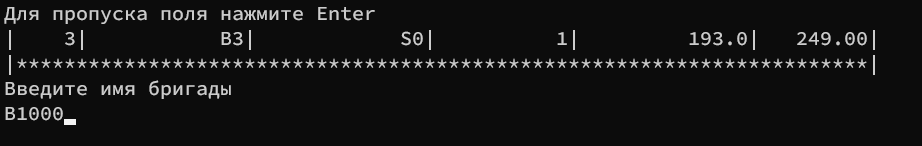
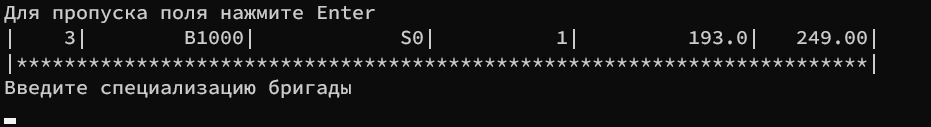


**Удаление элементов из списка:**

При удалении элемента из списка будет произведен поиск элемента в списке и выведена информация, после чего будет запрошено подтверждение удаления.

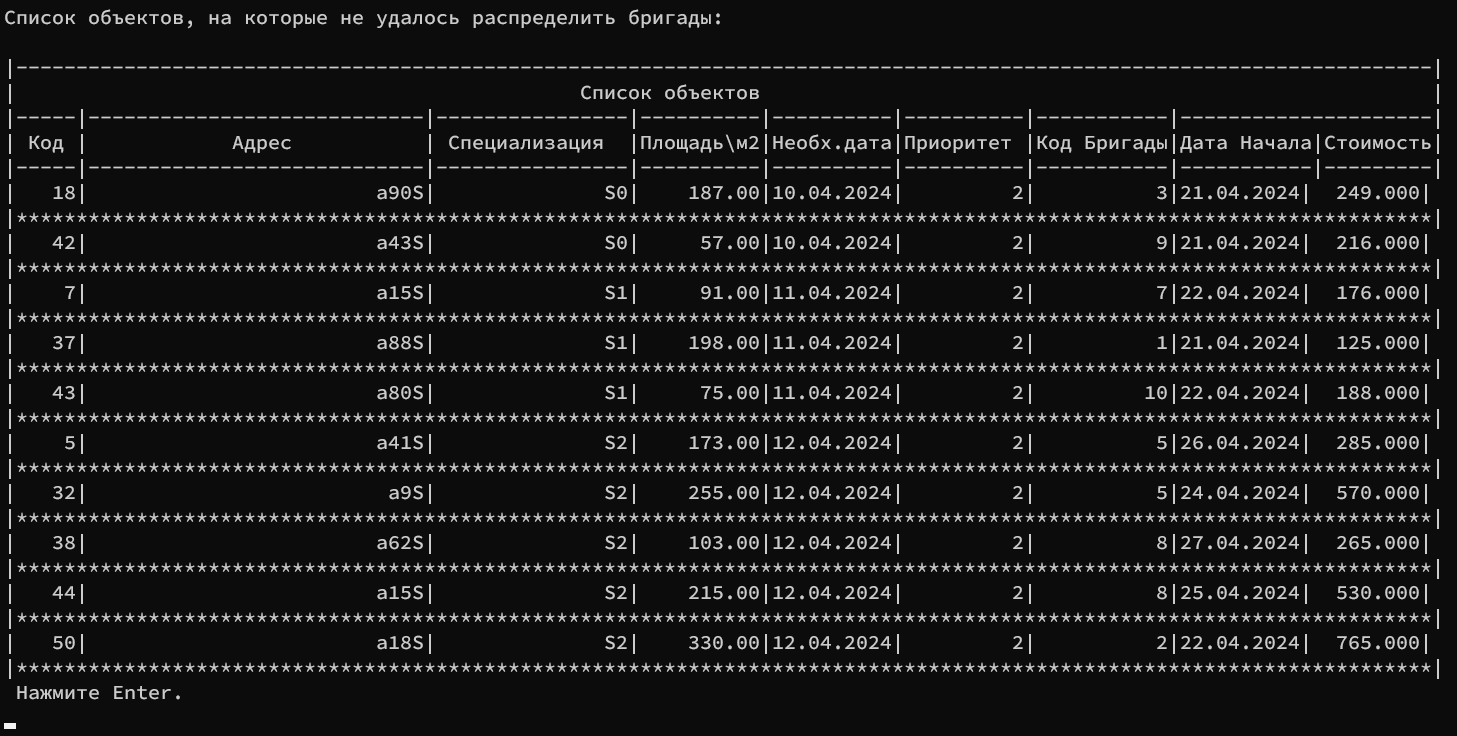
**Редактирование данных:**

При редактировании элемента будет выведено подменю выбора списка для редактирования. Затем информация, которая будет вводится в качестве данных элементов списка будет постепенно выведена на экран



**Распределение бригад по объектам:**

При выборе пункта распределения бригад по объектам будет выведен результат работы. В случае неудачи будут выведены объекты, на которые не получится распределить бригады, иначе будет выведено сообщение об успехе



**Сохранение:**

При работе с приложением возможно сохранение в файл для его последующего чтения.

**Важно!**

1.В течении одной сессии файл может быть прочитан один раз. В случае желания пользователя прочитать другой файл, необходимо завершить программу и прочитать нужный файл.

2. Если файл во время сессии не был прочтен, то при добавлении в списки будет выведено предупреждение о возможной потере текущей информации при последующем его чтении.

Все данные сохраненной сессии будут сохранены в папке Sessions, располагающейся в одной директории с данной программой.

При активации спец. функции результаты будут размещены в папке SpecFuncResults, находящейся рядом с программой.

**Обработка ошибок:**

В случае введения некорректных данных будет отображено соответствующее 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.

**2.15 Код программы**

program YP;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils,

Windows,

DateUtils,

IOUtils,

SpecialFunction in 'SpecialFunction.pas',

Sorts in 'Sorts.pas';

function IsValidNum(var value; const Id: TTypes; const purpOp: Integer)

: boolean; forward;

procedure SearchBrigElem(const brigHead: PtBrig); forward;

procedure SearchConstrElem(const constrHead: PtConstr); forward;

procedure SearchBrigElemByCode(var flagFound: boolean; const brigHead: PtBrig;

var brigTemp: PtBrig; var tempElem: PtComp); forward;

function InputFileName(): Tstring;

var

flagcheck: boolean;

function IsValidName(CheckStr: Tstring): boolean;

var

i: Integer;

begin

result := true;

if (CheckStr[1] = ' ') or (Length(CheckStr) = 0) then

result := false

else

for i := 1 to Length(CheckStr) do

begin

if (CheckStr[i] <= #31) or (CheckStr[i] = '<') or (CheckStr[i] = '>') or

(CheckStr[i] = '/') or (CheckStr[i] = '\') or (CheckStr[i] = '|') or

(CheckStr[i] = ':') or (CheckStr[i] = '?') or (CheckStr[i] = '\*') or

(CheckStr[i] = '"') then

result := false;

end;

if CheckStr[Length(CheckStr)] = ' ' then

result := false;

end;

begin

flagcheck := false;

Writeln('Введите название файла');

repeat

readln(result);

if not IsValidName(result) then

Writeln('Имя файла введено некорректно. Пожалуйста повторите попытку')

else

flagcheck := true;

until flagcheck;

end;

procedure OpenFileBrigForReading(var FBrigs: TBFile; const filePath: String);

begin

Assign(FBrigs, filePath + '\Brigades');

{$I-}

Reset(FBrigs);

{$I+}

If IOResult <> 0 Then

Rewrite(FBrigs);

end;

procedure OpenFileBrigForWriting(var FBrigs: TBFile; const filePath: String);

begin

Assign(FBrigs, filePath + '\Brigades');

Rewrite(FBrigs);

end;

procedure OpenFileConstrForWriting(var FConstr: TCFile; const filePath: String);

begin

Assign(FConstr, filePath + '\Constructions');

Rewrite(FConstr);

end;

procedure OpenFileConstrForReading(var FConstr: TCFile; const filePath: String);

begin

Assign(FConstr, filePath + '\Constructions');

{$I-}

Reset(FConstr);

{$I+}

If IOResult <> 0 Then

Rewrite(FConstr);

end;

procedure OpenFileSF(var FResultSf: TTFile; const filePath: String);

begin

Assign(FResultSf, filePath + '\Results');

Rewrite(FResultSf);

end;

//

procedure WriteInTextFile(const constrHead: PtConstr; const brigHead: PtBrig;

const FResultSf: TTFile);

var

tempConstr: PtConstr;

tempBrig: PtBrig;

flag: boolean;

begin

tempBrig := brigHead.next;

Writeln(FResultSf, 'Информация о распределении бригад по объектам':45);

Writeln(FResultSf, 'Дата: ' + DateToStr(Now) + ' ' + TimeToStr(Now):18);

while tempBrig <> nil do

begin

tempConstr := constrHead.next;

Writeln(FResultSf,

'----------------------------------------------------':51);

flag := false;

Writeln(FResultSf, 'Код бригады: ':30, tempBrig.brigInf.brigCode:21, '|');

Writeln(FResultSf, 'Имя бригады: ':30, tempBrig.brigInf.brigName:21, '|');

Writeln(FResultSf, 'Специализация бригады: ':30,

tempBrig.brigInf.brigSpec:21, '|');

Writeln(FResultSf, 'Количество рабочих бригады: ':30,

tempBrig.brigInf.brigMemCnt:21, '|');

Writeln(FResultSf, 'Выработка бригады в день: ':30,

tempBrig.brigInf.brigSqDev:21:4, '|');

Writeln(FResultSf, 'Стоимость дня работы бригады: ':30,

tempBrig.brigInf.brigPrice:21:4, '|');

Writeln(FResultSf,

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*':51);

Writeln(FResultSf, 'Информация о распределении данной бригады':51, '|');

Writeln(FResultSf,

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*':51);

Writeln(FResultSf,

'Список объектов, на которых работает данная бригада':51, '|');

Writeln(FResultSf,

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*':51);

while tempConstr <> nil do

begin

if tempConstr.constrInf.constrBrigadCode = tempBrig.brigInf.brigCode then

begin

flag := true;

Writeln(FResultSf, 'Код объекта: ':30,

tempConstr.constrInf.constrCode:21, '|');

Writeln(FResultSf, 'Адрес объекта: ':30,

tempConstr.constrInf.constrAddr:21, '|');

Writeln(FResultSf, 'Дата начала работы: ':30,

DateToStr(tempConstr.constrInf.constrStDate):21, '|');

Writeln(FResultSf, 'Стоимость всей работы: ':30,

tempConstr.constrInf.constrPrice:21:4, '|');

Writeln(FResultSf,

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*':50);

end;

tempConstr := tempConstr.next;

end;

if flag = false then

Writeln(FResultSf,

'Данная бригада не распределена ни на один объект':51, '|');

Writeln(FResultSf,

'----------------------------------------------------':51);

Writeln(FResultSf);

Writeln(FResultSf);

tempBrig := tempBrig.next;

end;

Close(FResultSf);

end;

procedure WriteInConstrFile(const constrHead: PtConstr; const FConstr: TCFile);

var

temp: PtConstr;

begin

temp := constrHead;

while temp <> nil do

begin

write(FConstr, temp.constrInf);

temp := temp.next;

end;

Close(FConstr);

end;

procedure WriteInBrigFile(const brigHead: PtBrig; const FBrig: TBFile);

var

temp: PtBrig;

begin

temp := brigHead;

while temp <> nil do

begin

write(FBrig, temp.brigInf);

temp := temp.next;

end;

Close(FBrig);

end;

procedure ReadInConstrList(var constrHead, constrTail: PtConstr;

const FConstr: TCFile);

var

temp: PtConstr;

begin

if FileSize(FConstr) <> 0 then

begin

read(FConstr, constrHead.constrInf);

temp := constrHead;

temp.next := nil;

while not Eof(FConstr) do

begin

new(temp.next);

read(FConstr, temp.next.constrInf);

temp := temp.next;

end;

temp.next := nil;

constrTail := temp;

end;

Close(FConstr);

end;

procedure ReadInBrigList(var brigHead: PtBrig; const FBrig: TBFile);

var

temp: PtBrig;

begin

if FileSize(FBrig) <> 0 then

begin

read(FBrig, brigHead.brigInf);

temp := brigHead;

temp.next := nil;

while not Eof(FBrig) do

begin

new(temp.next);

read(FBrig, temp.next.brigInf);

temp := temp.next;

end;

temp.next := nil;

end;

Close(FBrig);

end;

function IsDirectoryExist(const typeDirectory: Integer): boolean;

begin

case typeDirectory of

0:

begin

if not(DirectoryExists(GetCurrentDir() + '\Sessions')) then

begin

CreateDir(GetCurrentDir() + '\Sessions');

result := false;

end

else

result := true;

end;

1:

begin

if not(DirectoryExists(GetCurrentDir() + '\SpecFuncResults')) then

begin

CreateDir(GetCurrentDir() + '\SpecFuncResults');

result := false;

end

else

result := true;

end;

end;

end;

procedure ChooseSession(var realPath: String; var freadB, freadC: boolean;

const typeDirectory: Integer);

const

directoryNames: array [0 .. 1] of Tstring = ('\Sessions', '\SpecFuncResults');

var

foldersPaths: array of string;

resultNum: Integer;

resultStr: Tstring;

flagcheck: boolean;

path: String;

begin

resultNum := 0;

flagcheck := true;

SetLength(foldersPaths, Length(TDirectory.GetDirectories(GetCurrentDir() +

directoryNames[typeDirectory])));

for path in TDirectory.GetDirectories

((GetCurrentDir() + directoryNames[typeDirectory])) do

begin

foldersPaths[resultNum] := path;

Writeln(resultNum + 1, ' <--> ', ExtractFileName(path));

inc(resultNum);

end;

if (Length(foldersPaths) = 0) then

begin

Writeln('Файлов для чтения не существует.');

freadC := false;

freadB := false;

resultStr := InputFileName;

CreateDir(GetCurrentDir() + directoryNames[typeDirectory] + '\' +

resultStr);

realPath := GetCurrentDir() + directoryNames[typeDirectory] + '\' +

resultStr;

end

else

begin

Writeln('Выберите номер директории, с которой хотите работать или введите 0, чтобы создать новую');

while flagcheck do

begin

freadB := true;

freadC := true;

readln(resultStr);

if (resultStr >= '1') and (resultStr <= IntToStr(Length(foldersPaths)))

then

begin

flagcheck := false;

realPath := foldersPaths[StrToInt(resultStr) - 1];

end

else if (resultStr = '0') then

begin

resultStr := InputFileName;

CreateDir(GetCurrentDir() + directoryNames[typeDirectory] + '\' +

resultStr);

realPath := GetCurrentDir() + directoryNames[typeDirectory] + '\' +

resultStr;

Writeln('Данная информация была связана с директорией ', resultStr,

' в директории', copy(directoryNames[typeDirectory], 0,

Length(directoryNames[typeDirectory]) - 1));

flagcheck := false;

end

else

Writeln('Данные введены неверно. Повторите попытку');

end;

end;

end;

procedure CheckSessions(var realPath: String; var freadC, freadB: boolean;

const typeDirectory: Integer);

begin

if (IsDirectoryExist(typeDirectory)) or (typeDirectory=1) then

ChooseSession(realPath, freadB, freadC, typeDirectory);

end;

procedure OutputBrigHeadList;

begin

Writeln('|-----------------------------------------------------------------------|');

Writeln('| Список бригад |');

Writeln('|-----|-------------|--------------|-----------|--------------|---------|');

Writeln('| Код | Имя | Cпециализация|Кол-во раб.|Выраб. м2/день|Стоимость|');

Writeln('|-----|-------------|--------------|-----------|--------------|---------|');

end;

procedure OutputBrigList(const brigHead: PtBrig);

var

temp: PtBrig;

begin

temp := brigHead.next;

OutputBrigHeadList;

while temp <> nil do

begin

OutputElem(temp, brigadeData);

temp := temp.next;

end;

Writeln('|-----------------------------------------------------------------------|');

end;

procedure OutputConstrList(const constrHead: PtConstr);

var

temp: PtConstr;

begin

temp := constrHead.next;

OutputConstrHeadList;

while temp <> nil do

begin

OutputElem(temp, constrData);

temp := temp.next;

end;

Writeln('|----------------------------------------------------------------------------------------------------------------------|');

end;

procedure ClearScreen(const OriginX, OriginY: Integer; CountSize: Integer);

var

stdout: THandle;

csbi: TConsoleScreenBufferInfo;

ConsoleSize: DWORD;

NumWritten: DWORD;

Origin: TCoord;

begin

stdout := GetStdHandle(STD\_OUTPUT\_HANDLE);

Win32Check(stdout <> INVALID\_HANDLE\_VALUE);

Win32Check(GetConsoleScreenBufferInfo(stdout, csbi));

if CountSize <= 0 then

CountSize := csbi.dwSize.Y;

ConsoleSize := csbi.dwSize.X \* CountSize;

Origin.X := OriginX;

Origin.Y := OriginY;

Win32Check(FillConsoleOutputCharacter(stdout, ' ', ConsoleSize, Origin,

NumWritten));

Win32Check(SetConsoleCursorPosition(stdout, Origin));

end;

procedure InitLists(var brigadeHead: PtBrig;

var constrHead, constrTail: PtConstr);

begin

new(brigadeHead);

with brigadeHead^.brigInf do

begin

brigCode := 1;

brigName := '';

brigSpec := '';

brigMemCnt := 0;

brigSqDev := 0;

brigPrice := 0;

end;

brigadeHead.next := nil;

new(constrHead);

constrTail := constrHead;

with constrHead^.constrInf do

begin

constrCode := 1;

constrAddr := '';

constrSpec := '';

constrArea := 0;

constrStDateToDo := 0;

constrPriority := '';

constrBrigadCode := 0;

constrStDate := 0;

constrPrice := 0;

end;

constrHead.next := nil;

end;

function IsValDate(var tryString: Tstring; const purpOp: Integer): boolean;

var

err, i: Integer;

year, month, day: word;

begin

err := 0;

i := 1;

Trim(tryString);

result := false;

val(copy(tryString, 1, 2), day, err);

if (err = 0) and (tryString[3] = '.') and (tryString[6] = '.') then

begin

val(copy(tryString, 4, 2), month, err);

if err = 0 then

begin

val(copy(tryString, 7, 10), year, err);

if (err = 0) and IsValidDate(year, month, day) and (year >= 1900) then

result := true;

end;

end;

end;

procedure InputDate(var date: TdateTime; const purpOp: Integer);

var

tryString: Tstring;

flag: boolean;

begin

Writeln('Формат даты: дд.мм.гггг');

readln(tryString);

flag := false;

if not((purpOp = 1) and (tryString = '')) then

begin

while not((IsValDate(tryString, purpOp)) or (flag)) do

begin

Writeln('Дата введена неверно. Повторите попытку');

readln(tryString);

if (tryString = '') and (purpOp = 1) then

flag := true;

end;

if (flag = false) then

date := StrToDate(tryString);

end;

end;

function IsValidNum(var value; const Id: TTypes; const purpOp: Integer)

: boolean;

var

tryString: Tstring;

flag: boolean;

function ValElem: Integer;

var

tempInt,err: Integer;

tempRl:Real;

tempCurr:currency;

begin

case Id of

int:

begin

tempInt:=Integer(value);

val(tryString, Integer(value), err);

err := Ord((Integer(value) <= 0) or (err = 1));

if err<>0 then

Integer(value):=tempInt;

end;

curr:

begin

tempCurr:=Currency(Value);

val(tryString, Currency(value), err);

err := Ord((Currency(value) <= 0) or (err = 1));

if err<>0 then

Currency(value):=tempCurr;

end;

rl:

begin

temprl:=real(value);

val(tryString, Real(value), err);

err := Ord((Real(value) <= 0) or (err = 1));

if err<>0 then

Real(value):=tempRl;

end;

end;

result := err;

end;

begin

readln(tryString);

flag := true;

result := false;

if not((purpOp = 1) and (tryString = '')) then

while (flag) and (ValElem <> 0) do

begin

Writeln('Неверный ввод. Повторите попытку');

readln(tryString);

if ((purpOp = 1) and (tryString = '')) then

begin

result := true;

flag := false;

end;

end

else

result := true;

end;

procedure SelectOption(var fEndProc: boolean);

var

flagcheck: boolean;

resultString: string[10];

begin

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

fEndProc := false;

flagcheck := false;

end;

'0':

begin

ClearScreen(0, 0, -1);

fEndProc := true;

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

procedure CheckPriority(var resultString: Tstring; const purpOp: Integer);

var

flag: boolean;

begin

flag := false;

if (purpOp = 0) then

while not((resultString[1] <= '2') and (resultString[1] >= '0') and

(Length(resultString) < 2)) do

begin

Writeln('Данные введены неверно. Повторите попытку');

readln(resultString);

end

else

while not(flag) do

begin

if (resultString = '') or

((resultString[1] <= '2') and (resultString[1] >= '0') and

(Length(resultString) < 2)) then

flag := true

else

begin

Writeln('Данные введены неверно. Повторите попытку');

readln(resultString);

end;

end

end;

procedure InputInfoConstr(var newElem: PtConstr; var fStopInput: boolean;

const purpOp: Integer; const brigHead: PtBrig);

var

resultString: Tstring;

flagInput: boolean;

flagFound: boolean;

brigTemp: PtBrig;

begin

if purpOp = 0 then

begin

Writeln('Код объекта: ', newElem.constrInf.constrCode);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

OutputElem(newElem, constrData);

end;

Writeln('Введите адрес объекта');

if purpOp = 0 then

begin

readln(newElem.constrInf.constrAddr);

ClearScreen(0, 1, 3);

Writeln('Адрес объекта: ', newElem.constrInf.constrAddr);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

readln(resultString);

if resultString <> '' then

newElem.constrInf.constrAddr := resultString;

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

Writeln('Введите специализацию объекта');

if purpOp = 0 then

begin

readln(newElem.constrInf.constrSpec);

ClearScreen(0, 2, 3);

Writeln('Специализация объекта: ', newElem.constrInf.constrSpec);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

readln(resultString);

if resultString <> '' then

newElem.constrInf.constrSpec := resultString;

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

Writeln('Введите площадь объекта:');

IsValidNum(newElem.constrInf.constrArea, rl, purpOp);

if purpOp = 0 then

begin

ClearScreen(0, 3, -1);

Writeln('Площадь объекта: ', newElem.constrInf.constrArea:4:2);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

Writeln('Введите дату, с которой можно начинать работы ');

InputDate(newElem.constrInf.constrStDateToDo, purpOp);

if purpOp = 0 then

begin

ClearScreen(0, 4, -1);

Writeln('Необходимая дата начала работы на объекте: ',

DateToStr(newElem.constrInf.constrStDateToDo));

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

Writeln('Введите приоритет срочности объекта: ', #10, 'по убыванию 0 -- 2');

readln(resultString);

CheckPriority(resultString, purpOp);

if resultString<>'' then

newElem.constrInf.constrPriority := resultString;

if (purpOp = 0) then

begin

ClearScreen(0, 5, -1);

Writeln('Приоритет объекта: ', newElem.constrInf.constrPriority);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

if purpOp = 0 then

begin

ClearScreen(0, 6, 1);

newElem.constrInf.constrBrigadCode := 0;

Writeln('Код назначенной бригады: <<Пусто>>');

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

flagInput := true;

Writeln('Введите код назначенной бригады :');

flagFound := false;

while flagInput do

begin

brigTemp := brigHead.next;

if IsValidNum(newElem.constrInf.constrBrigadCode, int, purpOp) then

begin

flagInput := false;

newElem.constrInf.constrBrigadCode := -1;

end

else

begin

if brigtemp<>nil then

while (brigTemp.next <> nil) and (not flagFound) do

begin

if newElem.constrInf.constrBrigadCode = brigTemp.brigInf.brigCode then

flagFound := true

else

brigTemp := brigTemp.next;

end;

if not flagFound then

Writeln('Данной бригады не существует, повторите попытку.')

else

flagInput := false;

end;

end;

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

if purpOp = 0 then

begin

ClearScreen(0, 7, 1);

newElem.constrInf.constrStDate := 0;

Writeln('Фактическая дата начала работы: <<Пусто>>');

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

Writeln('Введите фактическую дату начала работ:');

InputDate(newElem.constrInf.constrStDate, purpOp);

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

if purpOp = 0 then

begin

ClearScreen(0, 8, 1);

newElem.constrInf.constrPrice := 0;

Writeln('Стоимость работы: <<Пусто>>');

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

Writeln('Вы хотите добавить данный объект в список', #10,

'1 -- Да 0 -- Нет');

SelectOption(fStopInput);

end

else

begin

Writeln('Введите стоимость работ:');

IsValidNum(newElem.constrInf.constrPrice, curr, purpOp);

ClearScreen(0, 1, -1);

OutputElem(newElem, constrData);

end;

end;

procedure InputInfoBrig(var newElem: PtBrig; var fStopInput: boolean;

const purpOp: Integer);

var

resultString: Tstring;

begin

if purpOp = 0 then

begin

Writeln('Код бригады: ', newElem.brigInf.brigCode);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

OutputElem(newElem, brigadeData);

Writeln('Введите имя бригады');

if purpOp = 0 then

begin

readln(newElem.brigInf.brigName);

ClearScreen(0, 1, 3);

Writeln('Имя бригады: ', newElem.brigInf.brigName);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

readln(resultString);

if resultString <> '' then

newElem.brigInf.brigName := resultString;

ClearScreen(0, 1, -1);

OutputElem(newElem, brigadeData);

end;

Writeln('Введите специализацию бригады');

if purpOp = 0 then

begin

readln(newElem.brigInf.brigSpec);

ClearScreen(0, 2, 3);

Writeln('Специализация бригады: ', newElem.brigInf.brigSpec);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

readln(resultString);

if resultString <> '' then

newElem.brigInf.brigSpec := resultString;

ClearScreen(0, 1, -1);

OutputElem(newElem, brigadeData);

end;

Writeln('Введите количество членов бригады');

if purpOp = 0 then

begin

IsValidNum(newElem.brigInf.brigMemCnt, int, 0);

ClearScreen(0, 3, -1);

Writeln('Количество членов бригады: ', newElem.brigInf.brigMemCnt);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

IsValidNum(newElem.brigInf.brigMemCnt, int, 1);

ClearScreen(0, 1, -1);

OutputElem(newElem, brigadeData);

end;

Writeln('Введите выработку бригады в м2/день');

if purpOp = 0 then

begin

IsValidNum(newElem.brigInf.brigSqDev, rl, 0);

ClearScreen(0, 4, -1);

Writeln('Выработка бригады в день: ', newElem.brigInf.brigSqDev:4:2);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

end

else

begin

IsValidNum(newElem.brigInf.brigSqDev, rl, 1);

ClearScreen(0, 1, -1);

OutputElem(newElem, brigadeData);

end;

Writeln('Введите стоимость дня работы бригады');

if purpOp = 0 then

begin

IsValidNum(newElem.brigInf.brigPrice, curr, 0);

ClearScreen(0, 5, -1);

Writeln('Стоимость дня работы: ', newElem.brigInf.brigPrice:4:2);

Writeln('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

Writeln('Вы хотите добавить данный объект в список', #10,

'1 -- Да 0 -- Нет');

SelectOption(fStopInput);

end

else

begin

IsValidNum(newElem.brigInf.brigPrice, curr, 1);

ClearScreen(0, 1, -1);

OutputElem(newElem, brigadeData);

end;

end;

procedure AddElemToBrigList(var brigadeHead: PtBrig);

var

temp: PtBrig;

newElem: PtBrig;

fStopInput: boolean;

begin

new(newElem);

repeat

newElem.brigInf.brigCode := brigadeHead.brigInf.brigCode;

InputInfoBrig(newElem, fStopInput, 0);

if not fStopInput then

begin

temp := brigadeHead.next;

new(brigadeHead.next);

brigadeHead.next.next := temp;

with brigadeHead.next.brigInf do

begin

brigCode := newElem.brigInf.brigCode;

brigName := (newElem.brigInf.brigName);

brigSpec := newElem.brigInf.brigSpec;

brigMemCnt := newElem.brigInf.brigMemCnt;

brigSqDev := newElem.brigInf.brigSqDev;

brigPrice := newElem.brigInf.brigPrice;

end;

inc(brigadeHead.brigInf.brigMemCnt);

inc(brigadeHead.brigInf.brigCode);

Writeln('Хотите продолжить?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

end

else

begin

Writeln('Хотите начать заново?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

end;

until fStopInput;

dispose(newElem);

end;

procedure AddElemToConstrList(const constrHead: PtConstr;

var constrTail: PtConstr; brigHead: PtBrig);

var

newElem: PtConstr;

fStopInput: boolean;

begin

new(newElem);

repeat

newElem.constrInf.constrCode := constrHead.constrInf.constrCode;

InputInfoConstr(newElem, fStopInput, 0, brigHead);

if not fStopInput then

begin

new(constrTail.next);

constrTail := constrTail.next;

constrTail.next := nil;

inc(constrHead.constrInf.constrBrigadCode);

inc(constrHead.constrInf.constrCode);

with constrTail.constrInf do

begin

constrCode := newElem.constrInf.constrCode;

constrAddr := newElem.constrInf.constrAddr;

constrSpec := newElem.constrInf.constrSpec;

constrArea := newElem.constrInf.constrArea;

constrStDateToDo := newElem.constrInf.constrStDateToDo;

constrPriority := newElem.constrInf.constrPriority;

constrBrigadCode := newElem.constrInf.constrBrigadCode;

constrStDate := newElem.constrInf.constrStDate;

constrPrice := newElem.constrInf.constrPrice;

end;

Writeln('Хотите продолжить?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

end

else

begin

Writeln('Хотите начать заново?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

end;

until fStopInput;

dispose(newElem);

end;

procedure InputSearchElem(var tempElem: PtComp; const Fields: TFields;

data: TListData);

begin

Writeln('Введите элемент, который хотите найти');

if (Fields = addr) or (Fields = name) or (Fields = spec) then

readln(tempElem.str)

else if (Fields = code) or (Fields = brigCode) or (Fields = memcnt) then

IsValidNum(tempElem.int, int, 0)

else if (Fields = area) or (Fields = sqDev) then

IsValidNum(tempElem.rl, rl, 0)

else if (Fields = stDate) or (Fields = stDateToDo) then

InputDate(tempElem.date, 0)

else

IsValidNum(tempElem.curr, curr, 0);

end;

procedure InitSearch(var Fields: TFields; var listType: TListData);

var

flagcheck: boolean;

resultString: Tstring;

begin

flagcheck := true;

case listType of

brigadeData:

begin

Writeln('Введите значение поля, с которым хотите произвести данную процедуру:');

Writeln('1 -- Код бригады', #10, '2 -- Имя бригады', #10,

'3 -- Специализация бригады', #10, '4 -- Стоимость работы бригады',

#10, '5 -- Количество работников бригады', #10,

'6 -- Выработка бригады');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

Fields := code;

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

Fields := name;

flagcheck := false;

end;

'3':

begin

ClearScreen(0, 0, -1);

Fields := spec;

flagcheck := false;

end;

'4':

begin

ClearScreen(0, 0, -1);

Fields := price;

flagcheck := false;

end;

'5':

begin

ClearScreen(0, 0, -1);

Fields := memcnt;

flagcheck := false;

end;

'6':

begin

ClearScreen(0, 0, -1);

Fields := sqDev;

flagcheck := false;

end;

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

constrData:

begin

Writeln('Введите значение поля, с которым хотите произвести данную процедуру');

Writeln('1 -- Код объекта', #10, '2 -- Адрес объекта', #10,

'3 -- Специализация объекта', #10, '4 -- Стоимость работы на объекте',

#10, '5 -- Площадь объекта', #10,

'6 -- Дата с которого можно начать работы', #10,

'7 -- Дата начала работы', #10, '8 -- Код назначенной бригады', #10,

'9 -- Приоритет работы');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

Fields := code;

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

Fields := addr;

flagcheck := false;

end;

'3':

begin

ClearScreen(0, 0, -1);

Fields := spec;

flagcheck := false;

end;

'4':

begin

ClearScreen(0, 0, -1);

Fields := price;

flagcheck := false;

end;

'5':

begin

ClearScreen(0, 0, -1);

Fields := area;

flagcheck := false;

end;

'6':

begin

ClearScreen(0, 0, -1);

Fields := stDateToDo;

flagcheck := false;

end;

'7':

begin

ClearScreen(0, 0, -1);

Fields := stDate;

flagcheck := false;

end;

'8':

begin

ClearScreen(0, 0, -1);

Fields := brigCode;

flagcheck := false;

end;

'9':

begin

ClearScreen(0, 0, -1);

Fields := prior;

flagcheck := false;

end;

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

end;

end;

procedure SearchBrigElem(const brigHead: PtBrig);

var

tempBrig: PtBrig;

tempElem: PtComp;

Fields: TFields;

flagFound: boolean;

data: TListData;

begin

new(tempElem);

data := brigadeData;

InitSearch(Fields, data);

InputSearchElem(tempElem, Fields, data);

flagFound := false;

tempBrig := brigHead;

OutputBrigHeadList;

while tempBrig.next <> nil do

begin

if CompareBrigElems(Fields, tempBrig.next, tempElem) = 0 then

begin

OutputElem(tempBrig.next, brigadeData);

flagFound := true;

end;

tempBrig := tempBrig.next;

end;

if not flagFound then

Writeln('| Элемент не найден |');

Writeln('|-----------------------------------------------------------------------|');

dispose(tempElem);

end;

procedure SearchConstrElem(const constrHead: PtConstr);

var

tempConstr: PtConstr;

tempElem: PtComp;

Fields: TFields;

flagFound: boolean;

data: TListData;

begin

new(tempElem);

data := constrData;

InitSearch(Fields, data);

InputSearchElem(tempElem, Fields, data);

flagFound := false;

tempConstr := constrHead;

OutputConstrHeadList;

while tempConstr.next <> nil do

begin

if CompareConstrElems(Fields, tempConstr.next, tempElem) = 0 then

begin

OutputElem(tempConstr.next, constrData);

flagFound := true;

end;

tempConstr := tempConstr.next;

end;

if not flagFound then

Writeln('| Элемент не найден |');

Writeln('|----------------------------------------------------------------------------------------------------------------------|');

dispose(tempElem);

end;

Procedure Search(const brigHead: PtBrig; const constrHead: PtConstr);

var

flagcheck: boolean;

resultString: Tstring;

begin

flagcheck := true;

Writeln('Введите список для поиска: ', #10, '1 -- Список бригад', #10,

'2 -- Список объектов');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

SearchBrigElem(brigHead);

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

SearchConstrElem(constrHead);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

procedure InitAdd(var brigHead: PtBrig; var constrHead: PtConstr;

var constrTail: PtConstr);

var

flagcheck: boolean;

resultString: Tstring;

begin

Writeln('Введите номер списка для добавления элементов: ', #10,

'1 -- Список бригад', #10, '2 -- Список объектов');

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

AddElemToBrigList(brigHead);

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

AddElemToConstrList(constrHead, constrTail, brigHead);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

procedure SearchConstrElemByCode(var flagFound: boolean;

const constrHead: PtConstr; var constrTemp: PtConstr; var tempElem: PtComp);

begin

constrTemp := constrHead;

Writeln('Введите код элемента :');

IsValidNum(tempElem.int, int, 0);

while (constrTemp.next <> nil) and (not flagFound) do

begin

if CompareConstrElems(code, constrTemp.next, tempElem) = 0 then

begin

OutputElem(constrTemp.next, constrData);

flagFound := true;

end

else

constrTemp := constrTemp.next;

end;

end;

procedure DeleteConstrElem(var constrHead: PtConstr; var ConstrTail:Ptconstr);

var

resultString: Tstring;

constrTemp, temp2: PtConstr;

fStopInput, flagcheck, flagFound: boolean;

tempElem: PtComp;

begin

new(tempElem);

flagcheck := true;

fStopInput := false;

while not fStopInput do

begin

flagFound := false;

constrTemp := constrHead;

SearchConstrElemByCode(flagFound, constrHead, constrTemp, tempElem);

ClearScreen(0, 0, -1);

if (not flagFound) then

begin

Writeln('Данный элемент невозможно удалить', #10, 'Enter-- Ок');

fStopInput := true;

readln;

end

else

begin

OutputElem(constrTemp.next, constrData);

Writeln('Уверены, что хотите удалить данный объект из списка?', #10,

'1 -- Да 0 -- Нет');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

dec(constrHead.constrInf.constrBrigadCode);

ClearScreen(0, 0, -1);

flagcheck := false;

temp2 := constrTemp.next;

if temp2.constrInf.constrCode = constrHead.constrInf.

constrCode - 1 then

dec(constrHead.constrInf.constrCode);

constrTemp.next := constrTemp.next.next;

if temp2=constrtail then

ConstrTail:=constrTemp;

dispose(temp2);

end;

'0':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

Writeln('Хотите продолжить?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

flagcheck := true;

end;

end;

dispose(tempElem);

end;

procedure SearchBrigElemByCode(var flagFound: boolean; const brigHead: PtBrig;

var brigTemp: PtBrig; var tempElem: PtComp);

begin

brigTemp := brigHead;

Writeln('Введите код элемента: ');

IsValidNum(tempElem.int, int, 0);

while (brigTemp.next <> nil) and (not flagFound) do

begin

if CompareBrigElems(code, brigTemp.next, tempElem) = 0 then

begin

flagFound := true;

end

else

brigTemp := brigTemp.next;

end;

end;

procedure DeleteBrigElem(var brigHead: PtBrig; const constrHead: PtConstr);

var

resultString: Tstring;

brigTemp, temp2: PtBrig;

fStopInput, flagcheck, flagFound, flagFound2: boolean;

tempElem: PtComp;

constrTemp: PtConstr;

begin

new(tempElem);

flagcheck := true;

fStopInput := false;

while not fStopInput do

begin

constrTemp := constrHead;

flagFound := false;

SearchBrigElemByCode(flagFound, brigHead, brigTemp, tempElem);

ClearScreen(0, 0, -1);

if (not flagFound) then

begin

Writeln('Данный элемент невозможно удалить', #10, 'Enter-- Ок');

fStopInput := true;

readln;

end

else

begin

OutputElem(brigTemp.next, brigadeData);

Writeln('Уверены, что хотите удалить данную бригаду из списка?', #10,

'1 -- Да 0 -- Нет');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

tempElem.int := brigTemp.next.brigInf.brigCode;

while (constrTemp.next <> nil) do

begin

if CompareConstrElems(brigCode, constrTemp.next, tempElem) = 0

then

begin

constrTemp.next.constrInf.constrBrigadCode := -1;

constrTemp.next.constrInf.constrStDate:=0;

constrTemp.next.constrInf.constrPrice:=0;

end;

constrTemp := constrTemp.next;

end;

dec(brigHead.brigInf.brigMemCnt);

ClearScreen(0, 0, -1);

flagcheck := false;

temp2 := brigTemp.next;

if temp2.brigInf.brigCode = brigHead.brigInf.brigCode - 1 then

dec(brigHead.brigInf.brigCode);

brigTemp.next := brigTemp.next.next;

dispose(temp2);

end;

'0':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

Writeln('Хотите продолжить?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

flagcheck := true;

end;

end;

dispose(tempElem);

end;

procedure InitDel(var brigHead: PtBrig; var constrHead: PtConstr;

var constrTail: PtConstr);

var

flagcheck: boolean;

resultString: Tstring;

begin

Writeln('Введите номер списка для удаления элементов: ', #10,

'1 -- Список бригад', #10, '2 -- Список объектов');

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

DeleteBrigElem(brigHead, constrHead);

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

DeleteConstrElem(constrHead,Constrtail);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

procedure InitShowLists(const brigHead: PtBrig; const constrHead: PtConstr);

var

resultString: Tstring;

flagcheck: boolean;

begin

Writeln('Введите номер списка для просмотра элементов: ', #10,

'1 -- Список бригад', #10, '2 -- Список объектов');

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

OutputBrigList(brigHead);

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

OutputConstrList(constrHead);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

Procedure SortBrigList(var brigHead: PtBrig; TypeSort: Integer);

var

runner1, runner2, temp, temp3, temp4: PtBrig;

Fields: TFields;

data: TListData;

temp2: PtComp;

begin

new(temp2);

data := brigadeData;

InitSearch(Fields, data);

runner1 := brigHead;

while runner1.next <> nil do

begin

runner2 := runner1.next;

temp := runner1;

InitSortElemBrig(temp2, Fields, runner1.next);

while runner2.next <> nil do

begin

if TypeSort = 1 then

begin

if CompareBrigElems(Fields, runner2.next, temp2) = -1 then

begin

temp := runner2;

InitSortElemBrig(temp2, Fields, temp.next);

end;

end

else if TypeSort = 0 then

begin

if CompareBrigElems(Fields, runner2.next, temp2) = 1 then

begin

temp := runner2;

InitSortElemBrig(temp2, Fields, temp.next);

end;

end;

runner2 := runner2.next;

end;

if (runner1 <> temp) then

begin

temp4 := runner1.next;

runner1.next := temp.next;

temp.next := temp.next.next;

runner1.next.next := temp4;

if brigHead.next = temp4 then

brigHead.next := runner1.next;

end;

runner1 := runner1.next;

end;

dispose(temp2);

end;

Procedure SortConstrList(var constrHead, constrTail: PtConstr;

TypeSort: Integer);

var

runner1, runner2, temp, temp3, temp4: PtConstr;

Fields: TFields;

data: TListData;

temp2: PtComp;

begin

new(temp2);

data := constrData;

InitSearch(Fields, data);

runner1 := constrHead;

while runner1.next <> nil do

begin

runner2 := runner1.next;

temp := runner1;

InitSortElemConstr(temp2, Fields, runner1.next);

while runner2.next <> nil do

begin

if TypeSort = 1 then

begin

if CompareConstrElems(Fields, runner2.next, temp2) = -1 then

begin

temp := runner2;

InitSortElemConstr(temp2, Fields, temp.next);

end;

end

else if (TypeSort = 0) then

begin

if CompareConstrElems(Fields, runner2.next, temp2) = 1 then

begin

temp := runner2;

InitSortElemConstr(temp2, Fields, temp.next);

end;

end;

runner2 := runner2.next;

end;

if (runner1 <> temp) then

begin

temp4 := runner1.next;

runner1.next := temp.next;

if temp.next = constrTail then

constrTail := temp4;

temp.next := temp.next.next;

runner1.next.next := temp4;

if constrHead.next = temp4 then

constrHead.next := runner1.next;

end;

runner1 := runner1.next;

end;

dispose(temp2);

end;

procedure Sort(var brigHead: PtBrig; var constrHead, constrTail: PtConstr);

var

flagcheck: boolean;

resultString: Tstring;

procedure InitTypeSort(const TypeData: TListData);

var

resultString: Tstring;

flagcheck: boolean;

begin

Writeln('Как вы хотите отсортировать список?', #10,

'1 -- по возрастанию 0 -- по убыванию');

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

if TypeData = brigadeData then

SortBrigList(brigHead, 1)

else

SortConstrList(constrHead, constrTail, 1);

flagcheck := false;

end;

'0':

begin

ClearScreen(0, 0, -1);

if TypeData = brigadeData then

SortBrigList(brigHead, 0)

else

SortConstrList(constrHead, constrTail, 0);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

begin

flagcheck := true;

Writeln('Введите список для сортировки: ', #10, '1 -- Список бригад', #10,

'2 -- Список объектов');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

InitTypeSort(brigadeData);

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

InitTypeSort(constrData);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

procedure EditBrigList(const brigHead: PtBrig);

var

flagFound, fStopInput, flagcheck: boolean;

tempElem, brigTemp: PtBrig;

compElem: PtComp;

resultString: Tstring;

begin

new(tempElem);

new(compElem);

flagcheck := true;

fStopInput := false;

while not fStopInput do

begin

flagFound := false;

SearchBrigElemByCode(flagFound, brigHead, brigTemp, compElem);

ClearScreen(0, 0, -1);

if flagFound then

begin

Writeln('Для пропуска поля нажмите Enter');

tempElem.brigInf := brigTemp.next.brigInf;

InputInfoBrig(tempElem, fStopInput, 1);

Writeln('Уверены, что хотите изменить параметры данной бригады?', #10,

'1 -- Да 0 -- Нет');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

brigTemp.next.brigInf := tempElem.brigInf;

end;

'0':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

Writeln('Хотите продолжить?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

flagcheck := true;

end

else

begin

ClearScreen(0, 0, -1);

Writeln('Данный элемент невозможно изменить', #10, 'Enter-- Ок');

fStopInput := true;

readln;

end

end;

dispose(tempElem);

dispose(compElem);

end;

procedure EditConstrList(const constrHead: PtConstr; const brigHead: PtBrig);

var

flagFound, fStopInput, flagcheck: boolean;

tempElem, constrTemp: PtConstr;

compElem: PtComp;

resultString: Tstring;

begin

new(tempElem);

new(compElem);

flagcheck := true;

fStopInput := false;

while not fStopInput do

begin

flagFound := false;

SearchConstrElemByCode(flagFound, constrHead, constrTemp, compElem);

ClearScreen(0, 0, -1);

if flagFound then

begin

Writeln('Для пропуска поля нажмите Enter');

tempElem.constrInf := constrTemp.next.constrInf;

InputInfoConstr(tempElem, fStopInput, 1, brigHead);

Writeln('Уверены, что хотите изменить параметры данного объекта?', #10,

'1 -- Да 0 -- Нет');

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

constrTemp.next.constrInf := tempElem.constrInf;

end;

'0':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

Writeln('Хотите продолжить?', #10, '1 -- Да 0 -- Нет');

SelectOption(fStopInput);

flagcheck := true;

end

else

begin

ClearScreen(0, 0, -1);

Writeln('Данный элемент невозможно изменить', #10, 'Enter-- Ок');

fStopInput := true;

readln;

end

end;

dispose(tempElem);

dispose(compElem);

end;

procedure InitEditLists(const brigHead: PtBrig; const constrHead: PtConstr);

var

resultString: Tstring;

flagcheck: boolean;

begin

Writeln('Введите номер списка для изменения элементов: ', #10,

'1 -- Список бригад', #10, '2 -- Список объектов');

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

EditBrigList(brigHead);

flagcheck := false;

end;

'2':

begin

ClearScreen(0, 0, -1);

EditConstrList(constrHead, brigHead);

flagcheck := false;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

procedure CreateMenu(var brigHead: PtBrig;

var constrHead, constrTail: PtConstr);

var

fstop, freadC, freadB, flagRead: boolean;

fileConstr: TCFile;

fileBrig: TBFile;

fileTSf: TTFile;

filePath: String;

procedure MenuManager(var fEndProc, flagRead: boolean);

var

resultString: Tstring;

flagcheck, fstopInput2: boolean;

begin

fstopInput2 := false;

Writeln('Выберите пункт меню: ');

flagcheck := true;

while flagcheck do

begin

readln(resultString);

if Length(resultString) = 1 then

begin

case resultString[1] of

'1':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

if (not flagRead) or not(freadB and freadC) then

begin

CheckSessions(filePath, freadC, freadB, 0);

OpenFileBrigForReading(fileBrig, filePath);

OpenFileConstrForReading(fileConstr, filePath);

ReadInConstrList(constrHead, constrTail, fileConstr);

ReadInBrigList(brigHead, fileBrig);

Writeln('Данные сессии успешно связаны. Нажмите Enter');

flagRead := true;

end

else

Writeln('Данные файлов уже прочитаны');

readln;

ClearScreen(0, 0, -1);

end;

'2':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

InitShowLists(brigHead, constrHead);

Writeln('Нажмите Enter');

readln;

ClearScreen(0, 0, -1);

end;

'3':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

Sort(brigHead, constrHead, constrTail);

ClearScreen(0, 0, -1);

end;

'4':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

Search(brigHead, constrHead);

readln;

ClearScreen(0, 0, -1);

end;

'5':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

if not flagRead then

begin

Writeln('Данные файлов не были прочтены.', #10,

'В случае попытки чтения из файла, добавленные данные могут быть перезаписаны. Продолжить?',

#10, '0 -- нет 1 -- да');

SelectOption(fstopInput2);

end;

if not fstopInput2 then

InitAdd(brigHead, constrHead, constrTail);

Writeln('Нажмите Enter');

ClearScreen(0, 0, -1);

end;

'6':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

InitDel(brigHead, constrHead, constrTail);

ClearScreen(0, 0, -1);

end;

'7':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

InitEditLists(brigHead, constrHead);

ClearScreen(0, 0, -1);

end;

'8':

begin

ClearScreen(0, 0, -1);

flagcheck := false;

if SpecialFunc(constrHead, brigHead) then

begin

CheckSessions(filePath, freadC, freadB, 1);

OpenFileSF(fileTSf, filePath);

WriteInTextFile(constrHead, brigHead, fileTSf);

end;

ClearScreen(0, 0, -1);

end;

'9':

begin

ClearScreen(0, 0, -1);

fEndProc := false;

flagcheck := false;

Writeln('Спасибо, что использовали данное приложение');

readln;

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end

else if copy(resultString, 1, 2) = '10' then

begin

ClearScreen(0, 0, -1);

flagcheck := false;

CheckSessions(filePath, freadC, freadB, 0);

OpenFileBrigForWriting(fileBrig, filePath);

OpenFileConstrForWriting(fileConstr, filePath);

WriteInConstrFile(constrHead, fileConstr);

WriteInBrigFile(brigHead, fileBrig);

fEndProc := false;

ClearScreen(0, 0, -1);

Writeln('Спасибо, что использовали данное приложение');

readln;

ClearScreen(0, 0, -1);

end

else

begin

Writeln('Неверный ввод. Повторите попытку');

end;

end;

end;

begin

freadC := false;

freadB := false;

flagRead := false;

fstop := true;

while fstop do

begin

Writeln('Меню: |');

Writeln('1.Прочитать данные из файла |');

Writeln('2.Просмотреть списки |');

Writeln('3.Сортировать данные |');

Writeln('4.Поиск данных |');

Writeln('5.Добавить данные в список |');

Writeln('6.Удалить данные из списка |');

Writeln('7.Редактировать данные |');

Writeln('8.Распределение бригад по объектам |');

Writeln('9.Выйти |');

Writeln('10.Сохранить и выйти |');

Writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

MenuManager(fstop, flagRead);

end;

end;

var

brigHead: PtBrig;

constrHead, constrTail: PtConstr;

startArr: TbrigArr;

begin

InitLists(brigHead, constrHead, constrTail);

CreateMenu(brigHead, constrHead, constrTail);

end.

**Unit SpecialFunction:**

unit SpecialFunction;

interface

uses System.SysUtils,

DateUtils, Sorts;

function SpecialFunc(const constrHead: ptConstr; const brigHead: ptBrig):boolean;

implementation

type

TDatesArr = array of TDate;

procedure InitStartArrays(const constrHead: ptConstr; const brigHead: ptBrig;

var ConstrArr: TConstrArr; var BrigArr: TBrigArr);

var

i: integer;

tempConstr: ptConstr;

tempBrig: ptBrig;

begin

SetLength(ConstrArr, constrHead.constrInf.constrBrigadCode);

SetLength(BrigArr, brigHead.brigInf.brigMemCnt);

tempConstr := constrHead.next;

i := 0;

while tempConstr <> nil do

begin

ConstrArr[i] := tempConstr;

inc(i);

tempConstr := tempConstr.next;

end;

tempBrig := brigHead.next;

i := 0;

while tempBrig <> nil do

begin

BrigArr[i] := tempBrig;

inc(i);

tempBrig := tempBrig.next;

end;

end;

procedure FindSpecBoarders(var arr; typeData: TlistData;

var startIndex, finishIndex: integer);

begin

case typeData of

brigadeData:

begin

while (finishIndex <> High(TConstrArr(arr))) and

(TConstrArr(arr)[finishIndex + 1].constrInf.constrSpec = TConstrArr

(arr)[startIndex].constrInf.constrSpec) do

inc(finishIndex);

end;

constrData:

begin

while (finishIndex <> High(TBrigArr(arr))) and

(TBrigArr(arr)[finishIndex + 1].brigInf.brigSpec = TBrigArr(arr)

[startIndex].brigInf.brigSpec) do

inc(finishIndex);

end;

end;

end;

Function FindSpec(var BrigArr: TBrigArr; const Spec: TString): integer;

var

i: integer;

flag: boolean;

begin

i := Low(BrigArr);

Result := -1;

flag := True;

for i := Low(BrigArr) to High(BrigArr) do

begin

if (BrigArr[i].brigInf.brigSpec = Spec) and (flag) then

begin

Result := i;

flag := false;

end;

end;

end;

Function RoundUpper(var op1, op2: real): integer;

begin

if Frac(op1 / op2) <> 0 then

Result := Trunc(op1 / op2) + 1

else

Result := Trunc(op1 / op2);

end;

procedure OutputResults(const ConstrArr: TConstrArr; const BrigArr: TBrigArr;

const table: array of integer; const startDates: TDatesArr);

var

i: integer;

begin

for i := Low(table) to High(table) do

begin

ConstrArr[i].constrInf.constrBrigadCode := BrigArr[table[i]]

.brigInf.brigCode;

ConstrArr[i].constrInf.constrStDate := startDates[i];

ConstrArr[i].constrInf.constrPrice :=

RoundUpper(ConstrArr[i].constrInf.constrArea,

BrigArr[table[i]].brigInf.brigSqDev) \* BrigArr[table[i]]

.brigInf.brigPrice;

end;

end;

procedure MainLogicSpec(var ConstrArr: TConstrArr; var BrigArr: TBrigArr);

var

tempStart, tempFin, tempBrig: TDate;

datesStart: TDatesArr;

table: Array Of integer;

i, j, k: integer;

tempsqDev: real;

function FindFinDate(const startDate: TDate;

const brigNum, constrNum: integer): TDate;

var

Date: TDate;

begin

Date := startDate;

Result := IncDay(Date, RoundUpper(ConstrArr[constrNum].constrInf.constrArea,

BrigArr[brigNum].brigInf.brigSqDev))

end;

function CheckIsProper(const tempDate: TDate;

const tablePos, currBrig: integer): boolean;

begin

Result := false;

if tempDate < datesStart[tablePos] then

Result := True

else if tempDate = datesStart[tablePos] then

begin

if (BrigArr[currBrig].brigInf.brigSqDev > BrigArr[table[tablePos]]

.brigInf.brigSqDev) then

begin

if FindFinDate(datesStart[tablePos], table[tablePos], tablePos) <>

FindFinDate(tempStart, currBrig, tablePos) then

Result := True;

end

else if (BrigArr[currBrig].brigInf.brigSqDev < BrigArr[table[tablePos]]

.brigInf.brigSqDev) then

begin

if FindFinDate(datesStart[tablePos], table[tablePos], tablePos)

= FindFinDate(tempStart, currBrig, tablePos) then

Result := True;

end

end;

end;

begin

SetLength(datesStart, length(ConstrArr));

SetLength(table, length(ConstrArr));

BrigArr := MergeSortBrig(BrigArr, area, 0);

ConstrArr := MergeSortConstr(ConstrArr, area, 0);

ConstrArr := MergeSortConstr(ConstrArr, StdatetoDo, 1);

ConstrArr := MergeSortConstr(ConstrArr, prior, 1);

for i := low(datesStart) to High(datesStart) do

datesStart[i] := 0;

for i := Low(ConstrArr) to High(ConstrArr) do

begin

table[i] := 0;

for j := Low(BrigArr) to High(BrigArr) do

begin

tempStart := ConstrArr[i].constrInf.constrStDateToDo;

for k := Low(table) to i - 1 do

begin

if (table[k] = j) and ((datesStart[k] < FindFinDate(tempStart, j, i)) or

((FindFinDate(datesStart[k], j, k) >= tempStart) and

(datesStart[k] < FindFinDate(tempStart, j, i)))) then

begin

tempStart := FindFinDate(datesStart[k], j, k);

end;

end;

if tempStart <= ConstrArr[i].constrInf.constrStDateToDo then

tempStart := ConstrArr[i].constrInf.constrStDateToDo;

if datesStart[i] = 0 then

begin

table[i] := j;

datesStart[i] := tempStart;

end

else

begin

if CheckIsProper(tempStart, i, j) then

begin

table[i] := j;

datesStart[i] := tempStart;

end

end;

end;

end;

OutputResults(ConstrArr, BrigArr, table, datesStart);

end;

procedure OutputNotRegistered(const ConstrArr: TConstrArr);

var

flag: boolean;

i: integer;

begin

flag := false;

for i := Low(ConstrArr) to High(ConstrArr) do

begin

if (ConstrArr[i].constrInf.constrBrigadCode = -1) or

(DaysBetween(ConstrArr[i].constrInf.constrStDate,

ConstrArr[i].constrInf.constrStDateToDo) >= 10) then

begin

flag := True;

end;

end;

if flag then

begin

writeln('Список объектов, на которые не удалось распределить бригады:');

writeln;

OutputConstrHeadList;

for i := Low(ConstrArr) to High(ConstrArr) do

begin

if (ConstrArr[i].constrInf.constrBrigadCode = -1) or

(DaysBetween(ConstrArr[i].constrInf.constrStDate,

ConstrArr[i].constrInf.constrStDateToDo) >= 10) then

begin

OutputElem(ConstrArr[i], constrData);

end;

end

end

else

writeln('Спецфункция успешно применена ко всем объектам.');

writeln(' Нажмите Enter. ');

readln;

end;

function SpecialFunc(const constrHead: ptConstr; const brigHead: ptBrig):boolean;

var

ConstrArrStart, ConstrArrMain: TConstrArr;

BrigArrStart, BrigArrMain: TBrigArr;

strtIndexC, finIndexC, strtIndexB, finIndexB: integer;

i: integer;

procedure InitMainArrays();

var

i: integer;

begin

for i := strtIndexB to finIndexB do

begin

BrigArrMain[i - strtIndexB] := BrigArrStart[i];

end;

for i := strtIndexC to finIndexC do

begin

ConstrArrMain[i - strtIndexC] := ConstrArrStart[i];

end;

end;

begin

InitStartArrays(constrHead, brigHead, ConstrArrStart, BrigArrStart);

if (length(BrigArrStart) = 0) or (length(ConstrArrStart) = 0) then

begin

writeln('Один из списков пуст, невозможно рапсределить бригады');

result:=false;

readln;

end

else

begin

BrigArrStart := MergeSortBrig(BrigArrStart, Spec, 1);

ConstrArrStart := MergeSortConstr(ConstrArrStart, Spec, 1);

strtIndexC := 0;

finIndexC := 0;

strtIndexB := 0;

finIndexB := 0;

while strtIndexC <= High(ConstrArrStart) do

begin;

FindSpecBoarders(ConstrArrStart, constrData, strtIndexC, finIndexC);

strtIndexB := FindSpec(BrigArrStart,

ConstrArrStart[strtIndexC].constrInf.constrSpec);

finIndexB := strtIndexB;

if strtIndexB <> -1 then

begin

FindSpecBoarders(BrigArrStart, brigadeData, strtIndexB, finIndexB);

SetLength(BrigArrMain, finIndexB - strtIndexB + 1);

SetLength(ConstrArrMain, finIndexC - strtIndexC + 1);

InitMainArrays();

MainLogicSpec(ConstrArrMain, BrigArrMain);

end;

strtIndexC := finIndexC + 1;

finIndexC := strtIndexC;

end;

result:=true;

end;

if (length(ConstrArrStart) <> 0) then

OutputNotRegistered(ConstrArrStart);

end;

end.

**Unit Sorts:**

unit SpecialFunction;

interface

uses System.SysUtils,

DateUtils, Sorts;

function SpecialFunc(const constrHead: ptConstr; const brigHead: ptBrig):boolean;

implementation

type

TDatesArr = array of TDate;

procedure InitStartArrays(const constrHead: ptConstr; const brigHead: ptBrig;

var ConstrArr: TConstrArr; var BrigArr: TBrigArr);

var

i: integer;

tempConstr: ptConstr;

tempBrig: ptBrig;

begin

SetLength(ConstrArr, constrHead.constrInf.constrBrigadCode);

SetLength(BrigArr, brigHead.brigInf.brigMemCnt);

tempConstr := constrHead.next;

i := 0;

while tempConstr <> nil do

begin

ConstrArr[i] := tempConstr;

inc(i);

tempConstr := tempConstr.next;

end;

tempBrig := brigHead.next;

i := 0;

while tempBrig <> nil do

begin

BrigArr[i] := tempBrig;

inc(i);

tempBrig := tempBrig.next;

end;

end;

procedure FindSpecBoarders(var arr; typeData: TlistData;

var startIndex, finishIndex: integer);

begin

case typeData of

brigadeData:

begin

while (finishIndex <> High(TConstrArr(arr))) and

(TConstrArr(arr)[finishIndex + 1].constrInf.constrSpec = TConstrArr

(arr)[startIndex].constrInf.constrSpec) do

inc(finishIndex);

end;

constrData:

begin

while (finishIndex <> High(TBrigArr(arr))) and

(TBrigArr(arr)[finishIndex + 1].brigInf.brigSpec = TBrigArr(arr)

[startIndex].brigInf.brigSpec) do

inc(finishIndex);

end;

end;

end;

Function FindSpec(var BrigArr: TBrigArr; const Spec: TString): integer;

var

i: integer;

flag: boolean;

begin

i := Low(BrigArr);

Result := -1;

flag := True;

for i := Low(BrigArr) to High(BrigArr) do

begin

if (BrigArr[i].brigInf.brigSpec = Spec) and (flag) then

begin

Result := i;

flag := false;

end;

end;

end;

Function RoundUpper(var op1, op2: real): integer;

begin

if Frac(op1 / op2) <> 0 then

Result := Trunc(op1 / op2) + 1

else

Result := Trunc(op1 / op2);

end;

procedure OutputResults(const ConstrArr: TConstrArr; const BrigArr: TBrigArr;

const table: array of integer; const startDates: TDatesArr);

var

i: integer;

begin

for i := Low(table) to High(table) do

begin

ConstrArr[i].constrInf.constrBrigadCode := BrigArr[table[i]]

.brigInf.brigCode;

ConstrArr[i].constrInf.constrStDate := startDates[i];

ConstrArr[i].constrInf.constrPrice :=

RoundUpper(ConstrArr[i].constrInf.constrArea,

BrigArr[table[i]].brigInf.brigSqDev) \* BrigArr[table[i]]

.brigInf.brigPrice;

end;

end;

procedure MainLogicSpec(var ConstrArr: TConstrArr; var BrigArr: TBrigArr);

var

tempStart, tempFin, tempBrig: TDate;

datesStart: TDatesArr;

table: Array Of integer;

i, j, k: integer;

tempsqDev: real;

function FindFinDate(const startDate: TDate;

const brigNum, constrNum: integer): TDate;

var

Date: TDate;

begin

Date := startDate;

Result := IncDay(Date, RoundUpper(ConstrArr[constrNum].constrInf.constrArea,

BrigArr[brigNum].brigInf.brigSqDev))

end;

function CheckIsProper(const tempDate: TDate;

const tablePos, currBrig: integer): boolean;

begin

Result := false;

if tempDate < datesStart[tablePos] then

Result := True

else if tempDate = datesStart[tablePos] then

begin

if (BrigArr[currBrig].brigInf.brigSqDev > BrigArr[table[tablePos]]

.brigInf.brigSqDev) then

begin

if FindFinDate(datesStart[tablePos], table[tablePos], tablePos) <>

FindFinDate(tempStart, currBrig, tablePos) then

Result := True;

end

else if (BrigArr[currBrig].brigInf.brigSqDev < BrigArr[table[tablePos]]

.brigInf.brigSqDev) then

begin

if FindFinDate(datesStart[tablePos], table[tablePos], tablePos)

= FindFinDate(tempStart, currBrig, tablePos) then

Result := True;

end

end;

end;

begin

SetLength(datesStart, length(ConstrArr));

SetLength(table, length(ConstrArr));

BrigArr := MergeSortBrig(BrigArr, area, 0);

ConstrArr := MergeSortConstr(ConstrArr, area, 0);

ConstrArr := MergeSortConstr(ConstrArr, StdatetoDo, 1);

ConstrArr := MergeSortConstr(ConstrArr, prior, 1);

for i := low(datesStart) to High(datesStart) do

datesStart[i] := 0;

for i := Low(ConstrArr) to High(ConstrArr) do

begin

table[i] := 0;

for j := Low(BrigArr) to High(BrigArr) do

begin

tempStart := ConstrArr[i].constrInf.constrStDateToDo;

for k := Low(table) to i - 1 do

begin

if (table[k] = j) and ((datesStart[k] < FindFinDate(tempStart, j, i)) or

((FindFinDate(datesStart[k], j, k) >= tempStart) and

(datesStart[k] < FindFinDate(tempStart, j, i)))) then

begin

tempStart := FindFinDate(datesStart[k], j, k);

end;

end;

if tempStart <= ConstrArr[i].constrInf.constrStDateToDo then

tempStart := ConstrArr[i].constrInf.constrStDateToDo;

if datesStart[i] = 0 then

begin

table[i] := j;

datesStart[i] := tempStart;

end

else

begin

if CheckIsProper(tempStart, i, j) then

begin

table[i] := j;

datesStart[i] := tempStart;

end

end;

end;

end;

OutputResults(ConstrArr, BrigArr, table, datesStart);

end;

procedure OutputNotRegistered(const ConstrArr: TConstrArr);

var

flag: boolean;

i: integer;

begin

flag := false;

for i := Low(ConstrArr) to High(ConstrArr) do

begin

if (ConstrArr[i].constrInf.constrBrigadCode = -1) or

(DaysBetween(ConstrArr[i].constrInf.constrStDate,

ConstrArr[i].constrInf.constrStDateToDo) >= 10) then

begin

flag := True;

end;

end;

if flag then

begin

writeln('Список объектов, на которые не удалось распределить бригады:');

writeln;

OutputConstrHeadList;

for i := Low(ConstrArr) to High(ConstrArr) do

begin

if (ConstrArr[i].constrInf.constrBrigadCode = -1) or

(DaysBetween(ConstrArr[i].constrInf.constrStDate,

ConstrArr[i].constrInf.constrStDateToDo) >= 10) then

begin

OutputElem(ConstrArr[i], constrData);

end;

end

end

else

writeln('Спецфункция успешно применена ко всем объектам.');

writeln(' Нажмите Enter. ');

readln;

end;

function SpecialFunc(const constrHead: ptConstr; const brigHead: ptBrig):boolean;

var

ConstrArrStart, ConstrArrMain: TConstrArr;

BrigArrStart, BrigArrMain: TBrigArr;

strtIndexC, finIndexC, strtIndexB, finIndexB: integer;

i: integer;

procedure InitMainArrays();

var

i: integer;

begin

for i := strtIndexB to finIndexB do

begin

BrigArrMain[i - strtIndexB] := BrigArrStart[i];

end;

for i := strtIndexC to finIndexC do

begin

ConstrArrMain[i - strtIndexC] := ConstrArrStart[i];

end;

end;

begin

InitStartArrays(constrHead, brigHead, ConstrArrStart, BrigArrStart);

if (length(BrigArrStart) = 0) or (length(ConstrArrStart) = 0) then

begin

writeln('Один из списков пуст, невозможно рапсределить бригады');

result:=false;

readln;

end

else

begin

BrigArrStart := MergeSortBrig(BrigArrStart, Spec, 1);

ConstrArrStart := MergeSortConstr(ConstrArrStart, Spec, 1);

strtIndexC := 0;

finIndexC := 0;

strtIndexB := 0;

finIndexB := 0;

while strtIndexC <= High(ConstrArrStart) do

begin;

FindSpecBoarders(ConstrArrStart, constrData, strtIndexC, finIndexC);

strtIndexB := FindSpec(BrigArrStart,

ConstrArrStart[strtIndexC].constrInf.constrSpec);

finIndexB := strtIndexB;

if strtIndexB <> -1 then

begin

FindSpecBoarders(BrigArrStart, brigadeData, strtIndexB, finIndexB);

SetLength(BrigArrMain, finIndexB - strtIndexB + 1);

SetLength(ConstrArrMain, finIndexC - strtIndexC + 1);

InitMainArrays();

MainLogicSpec(ConstrArrMain, BrigArrMain);

end;

strtIndexC := finIndexC + 1;

finIndexC := strtIndexC;

end;

result:=true;

end;

if (length(ConstrArrStart) <> 0) then

OutputNotRegistered(ConstrArrStart);

end;

end.

**ПРИЛОЖЕНИЕ А**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

|  |  |
| --- | --- |
| **УТВЕРЖДАЮ**  Руководитель  Фадеева Елена Евгеньевна  «11» марта 2024 г. |  |
| Информационная система учета строительных работы  “Construction Manager”  наименование вида ИС  Construction Manager  Сокращенное наименование ИС  Техническое задание  Действует с 10.03.2024 г. | |
| **СОГЛАСОВАНО**  Руководитель  Фадеева Елена  Евгеньевна  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.2024 г | **РАЗРАБОТЧИК**  Студент  Радецкий Алексей Николаевич  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г. |
| Минск 2024 | |

**СОДЕРЖАНИЕ**

**1.Общие сведения60**

1.1. Наименование и реквизиты Заказчика60

1.2. Наименование и реквизиты Исполнителя60

1.3. Основание для разработки………………………………………………………………………………………………….60

1.4. Плановые сроки начала и окончания работ.............................................................................60

**2.НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ61**

2.1. Назначение Системы61

2.2. Цели создания Системы61

**3.** **ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ61**

3.1. Краткие сведения об объектах автоматизации61

3.2. Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды62

**4.TРЕБОВАНИЯ К СИСТЕМЕ62**

4.1. Требования к системе в целом62

4.1.1.1. Перечень подсистем, их назначение и основные характеристики62

4.1.1.2. Требования к организации обмена информацией между компонентами Системы.63

4.2. Функциональные требования63

4.3. Обработка ошибок64

4.3.1. Ошибки загрузки данных из внешних источников64

4.3.2. Внутренние ошибки.64

4.4. Интерфейс64

4.4.1. Основные требования65

# **1.ОБЩИЕ СВЕДЕНИЯ**

## Полное наименование Системы и её условное обозначение

Полное наименование системы: Информационная система учета строительных работы “Construction Manager”

*Краткое наименование системы: “СM”*

**1.1. Наименование и реквизиты Заказчика**

Заказчиком Системы является кафедра ПОИТ (ассистент Фадеева Елена Евгеньевна)

Контактный телефон: +375 44 552 37 38

**1.2. Наименование и реквизиты Исполнителя**

Исполнителем является (проектный менеджер – Радецкий Алексей Николаевич)

Адрес электронной почты: lesharad2006@gmail.com

Телефон: +375 44 757 50 44

**1.3. Основание для разработки**

- Список индивидуальных заданий УП

- Список вариантов к списку индивидуальных заданий

**1.4. Плановые сроки начала и окончания работ**

Начало разработки: 10 марта 2024 года;

Ожидаемое завершение разработки: 31 мая 2024 года

**2.НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ**

## 2.1. Назначение Системы

*Система “Construction Manager” (“СМ”) предназначена для создания приложения, способствующего упрощению ведения учета информации о строительных работах, а также ее активному оперированию.*

## Цели создания Системы

## Система " Construction Manager " (СМ) создана с целью частичной автоматизации деятельности администраторов строительной деятельности.

1. **ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ**

## Краткие сведения об объектах автоматизации

*В соответствии с пунктом 2.1 можно выделить следующие объекты автоматизации:*

1. Список бригад:

Каждый элемент данного списка содержит следующую информацию:

* Код бригады;
* Имя Бригады;
* Специализация бригады;
* Количество рабочих;
* Выработка бригады м2/день;
* Стоимость дня работы бригады;

1. Список строительных объектов:

Каждый элемент данного списка содержит следующую информацию:

* Код объекта;
* Адрес;
* Специализация работы;
* Площадь в м2;
* Дата, с которой можно начинать работы;
* Приоритет срочности;
* Код бригады;
* Дата начала работ;

## Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды

Планируется использовать систему в строительных организациях.

В основной круг пользователей, предполагается, будут входить менеджеры строительных организаций.

1. **TРЕБОВАНИЯ К СИСТЕМЕ**

## Требования к системе в целом

1. Пользователям приложения должны быть доступны следующие функции:

* Поиск информации по различным видам (например, код бригады, стоимость работ и дата начала работ)
* Просмотр актуальной информации о работе бригад.
* Добавление пользователями иной информации в списки.

#### Перечень подсистем, их назначение и основные характеристики

Программа должна содержать пользовательское меню, состоящее из следующих пунктов:

1.Чтение данных из файла

2.Просмотр всего списка (если несколько списков – выпадает подменю с именами списков для просмотра)

3.Сортировка данных в соответствии с заданием

4.Поиск данных с использованием фильтров

5.Добавление данных в список (если несколько списков – выпадает подменю с именами списков для добавления)

6.Удаление данных из списка (если несколько списков – выпадает подменю с именами списков для удаления)

7.Редактирование данных (если несколько списков – выпадает подменю с именами списков для редактирования)

8.Специальные функции задания

9.Выход из программы без сохранения изменений

10.Выход с сохранением изменений

#### Требования к организации обмена информацией между компонентами Системы

**1.Стандартизация данных:** Все компоненты системы должны использовать единые стандарты для представления данных. Это включает в себя стандартизированные форматы данных, соглашения об именовании полей и другие соглашения, которые облегчают интеграцию

**2**. **Интеграция с внешними источниками**: Система должна быть способной интегрировать данные из внешних источников, таких как сторонние файлы, базы данных.

## Функциональные требования

1. *Подсистемы организации работы меню и подменю:*

*Задачи:*

* *Отрисовка соответствующего меню/подменю;*
* *Получение данных от пользователя;*
* *Вызов подсистемы в соответствии с введёнными данными с передачей ей необходимых для работы данных;*

1. *Подсистемы работы с типизированными и текстовыми файлами:*

*Задачи:*

* *Проверка наличия соответствующих файлов;*
* *Чтение информации из типизированного/текстового файла;*
* *Запись информации в типизированный/текстовый файл;*

1. *Подсистемы просмотра выбранного списка:*

*Задачи:*

* *Обработка полученных при вызове данных;*
* *Вывод в консоль выбранного списка;*

1. *Подсистемы добавления/удаления/редактирования записей в выбранном списке:*

*Задачи:*

* *Обработка полученных при вызове данных;*
* *Изменение выбранного списка (добавление/удаление/редактирование) в соотв. с обработанными данными;*

1. *Подсистемы поиска записей по выбранному полю:*

*Задачи:*

* *Обработка полученных при вызове данных;*
* *Осуществление поиска в списке в соотв. с обработанными данными;*

1. *Подсистемы сортировки выбранного списка по определённому критерию (критерием может выступать поле, одним из критериев является суммарная стоимость работы):*

*Задачи:*

* *Обработка полученных при вызове данных;*
* *Сортировка выбранного списка в соотв. с обработанными данными;*

1. *Распределение объектов по бригадам (критерии описаны в пункте 3.1):*

*Задача:*

* *Обработка полученных при вызове данных;*
* *Осуществление корректной работы алгоритма по распределению объектов.*

## Обработка ошибок

### Ошибки загрузки данных из внешних источников

*Ошибки загрузки данных из внешних источников могут возникнуть в следующих случаях:*

1. *Повреждение рабочих файлов в результате некорректной работы Системы при прошлом запуске;*
2. *Повреждение рабочих файлов в результате непреднамеренных действий пользователя;*
3. *Изменение формата/имени рабочих файлов.*

*При возникновении ошибок данного типа необходимо уведомить пользователя и с его подтверждения создать новые рабочие файлы.*

### Внутренние ошибки

*Внутренние ошибки могут возникнуть в следующих случаях:*

1. *Ввод пользователем непредвиденных/некорректных данных. Для избежания подобного рода ошибок необходимо проводить проверку всех введённых пользователем данных. Также необходимо оповестить пользователя и позволить повторный ввод данных;*
2. *Некорректная работа с динамическими структурами данных. Результат подобного рода ошибок непредсказуем.*

## Интерфейс

**Пользовательский интерфейс:** должен быть интуитивно понятным и удобным для пользователей. Разрабатывается простой, но информативный интерфейс, который обеспечивает легкий доступ к основным функциям системы.

* + 1. **Основные требования**

Интерфейс должен быть реализован в консольном режиме.

Пользовательское меню должно содержать следующие пункты:

1. Прочитать данные из файла.;
2. Просмотреть списки.;
3. Сортировать данные.;
4. Найти данные в списках.;
5. Добавить данные в списки;
6. Удалить данные из списков.;
7. Редактировать списки.;
8. Создать Playlist.;
9. Выход из программы без сохранения изменений;
10. Выход с сохранением изменений.

Для пунктов №2, 3, 4, 5, 6, 7, 8 необходимо реализовать выпадающие подменю для ввода дополнительной информации (фильтры поиска, код удаляемой записи, интересующий список и т.д.).

***СОСТАВИЛИ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Наименование организации, предприятия*** | ***Должность исполнителя*** | ***Фамилия, имя, отчество*** | ***Подпись*** | ***Дата*** |
|  |  |  |  |  |

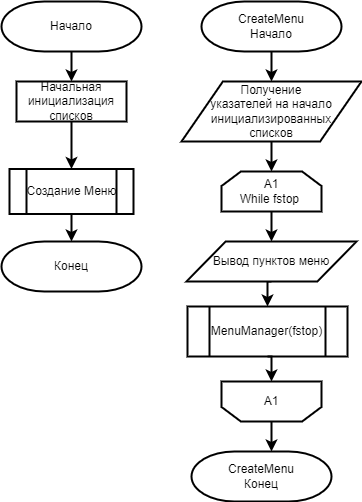
***СОГЛАСОВАНО***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Наименование организации, предприятия*** | ***Должность исполнителя*** | ***Фамилия, имя, отчество*** | ***Подпись*** | ***Дата*** |
|  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |

**ПРИЛОЖЕНИЕ Б**

**СХЕМЫ АЛГОРИТМОВ**

Схема Б.1 – Схема работы программы

****

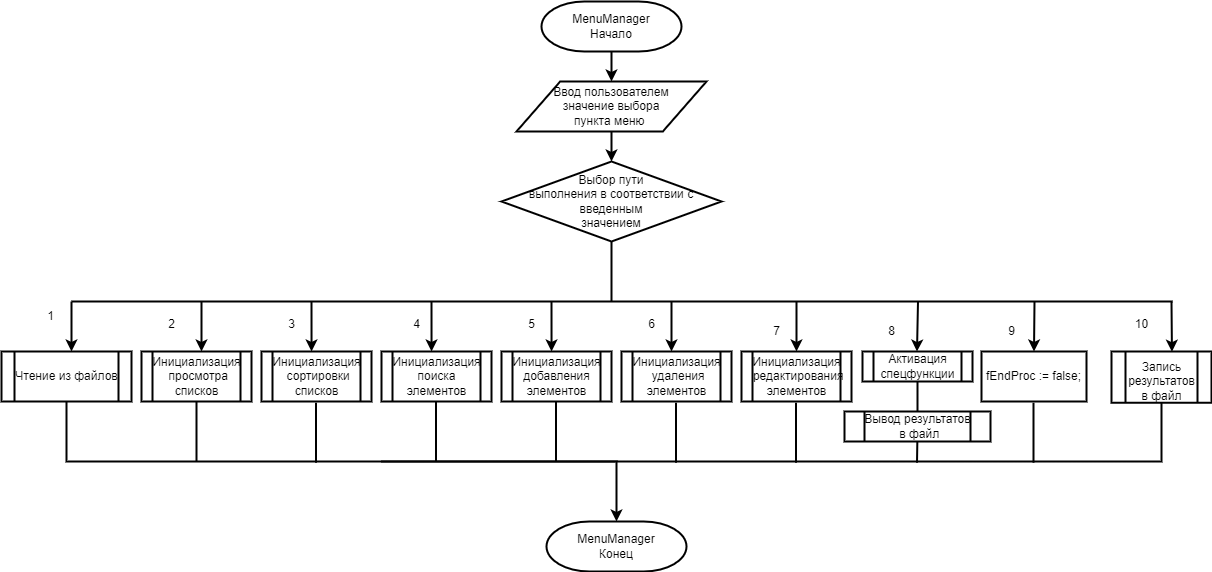
****Схема Б.2 – Схема работы программы

Схема Б.3 – Схема алгоритма чтения из файлов

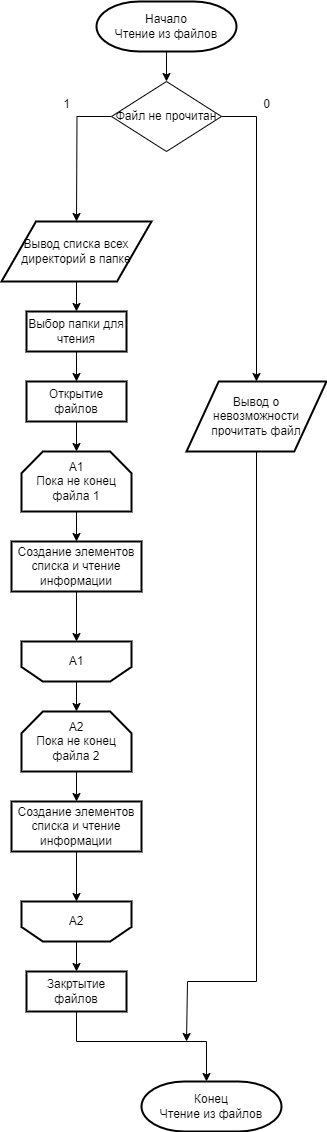
****

Схема Б.4 – Схема алгоритма инициализации вывода списков

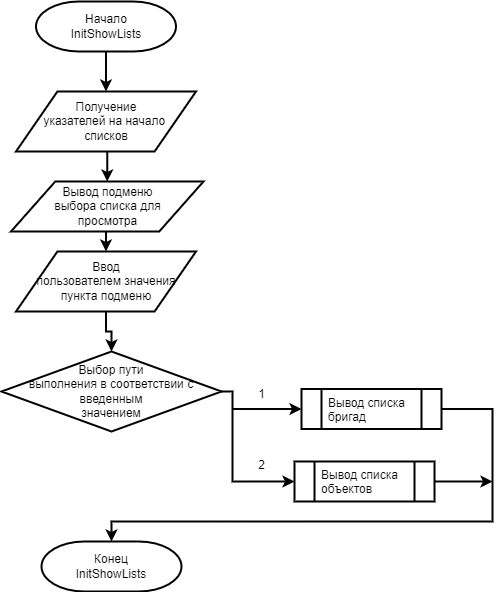
****

Схема Б.5 – Схема алгоритма вывода списка

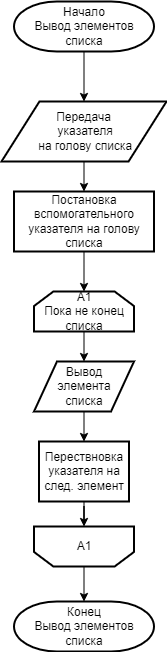


Схема Б.6 – Схема алгоритма добавления в список

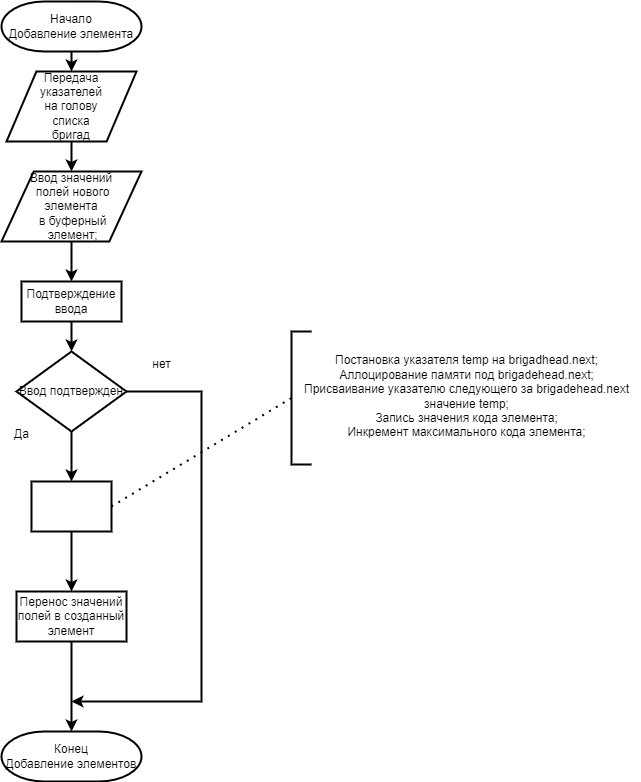
****

Схема Б.7 – Схема алгоритма поиска с фильтром

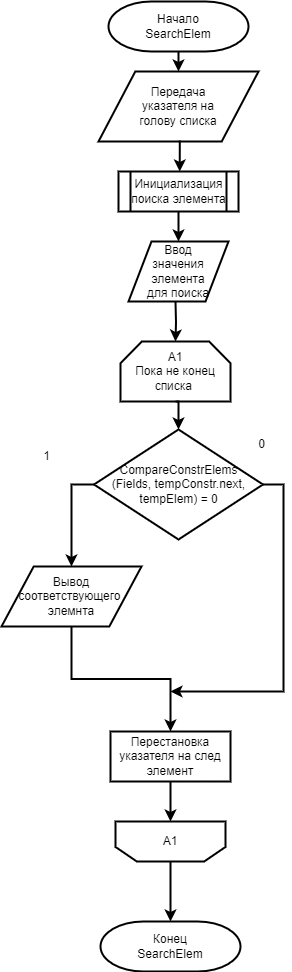
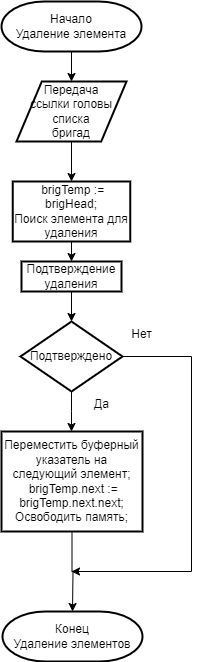
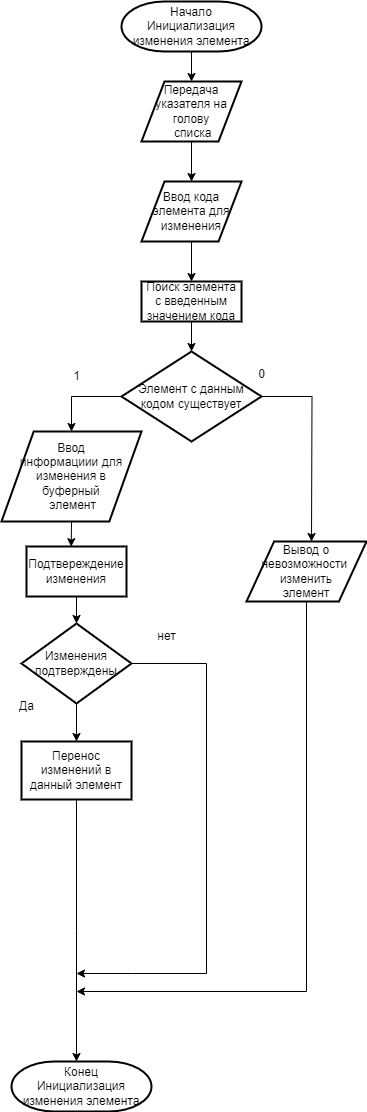


Схема Б.8 – Схема алгоритма удаления элемента из списка



Схема Б.9 – Схема алгоритма редактирования списка

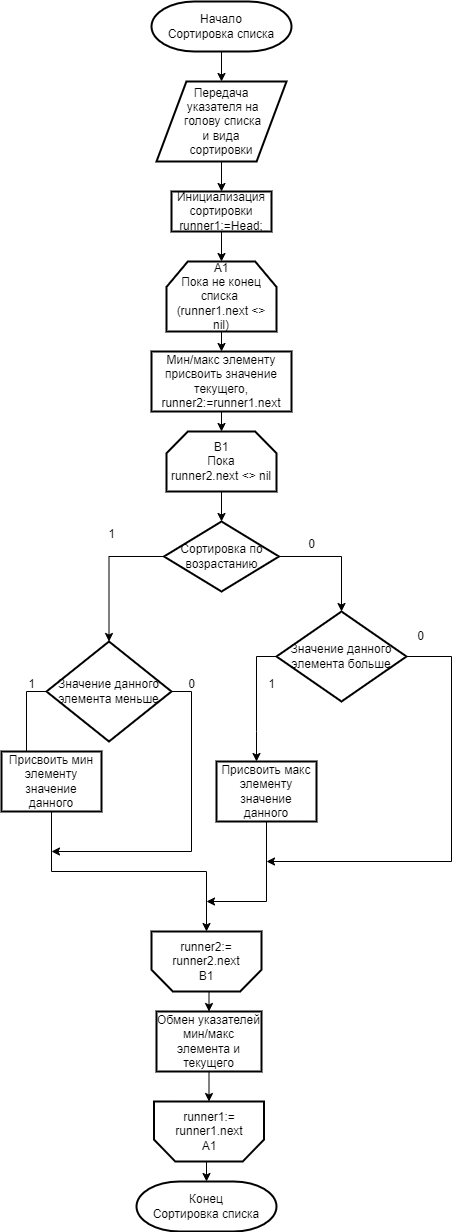
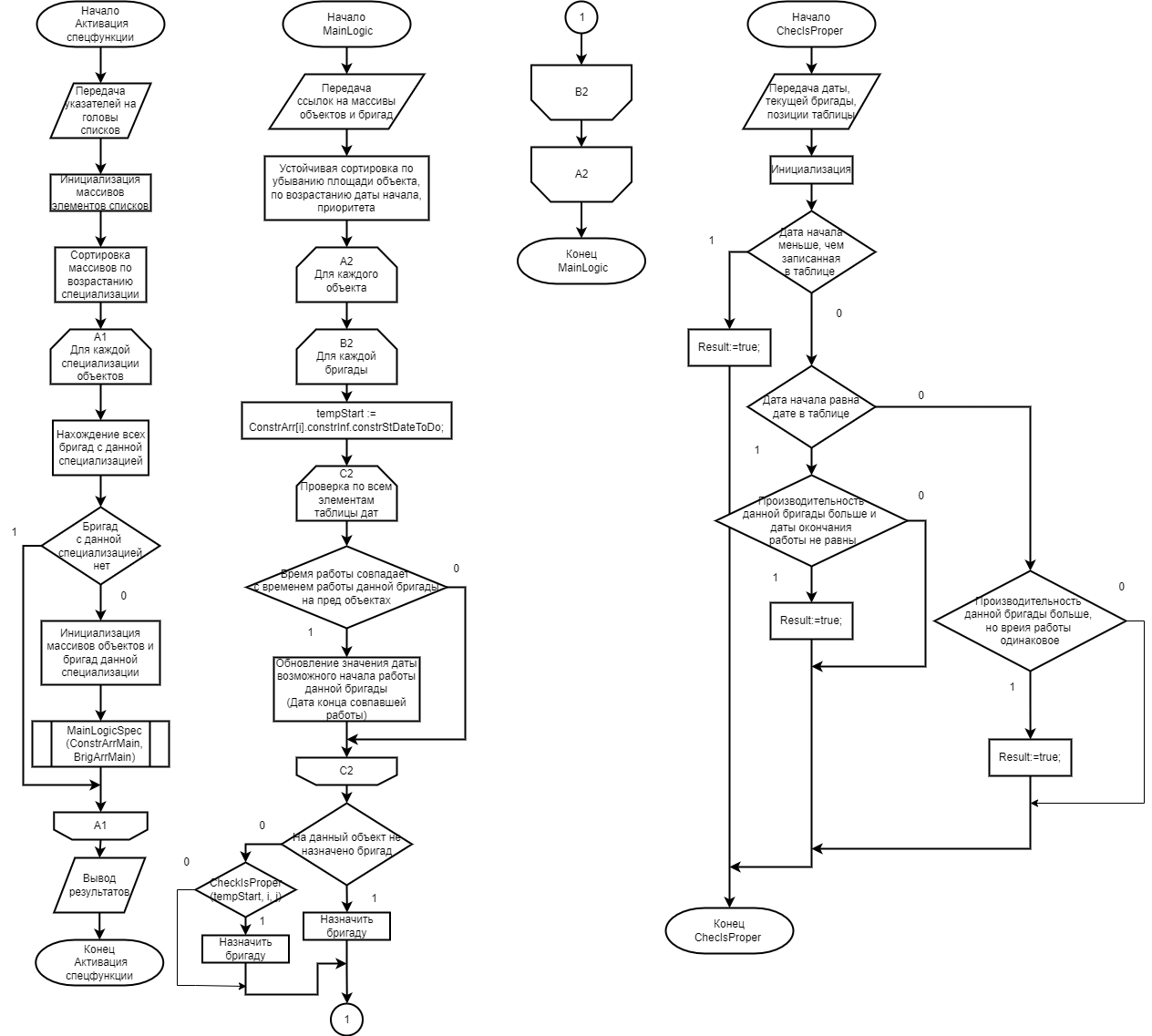
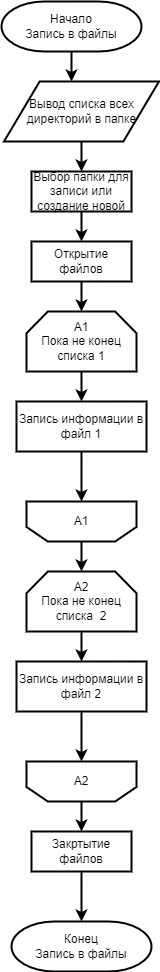
Схема Б.10 – Схема алгоритма сортировки списка

Схема Б.11 – Схема алгоритма специальной функции

****

Схема Б.12 – Схема алгоритма сохранения в файл