Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования(ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к учебной практике(ознакомительной)
на тему

КАТАЛОГ КОМПЬЮТЕРОВ

БГУИР КР 1-40 01 01 106 ПЗ

Студент К. А. Зиновенко
Руководитель асс. Е. Е. Фадеева

Минск, 2024

СОДЕРЖАНИЕ

1. Разработка технического задания.

Для программного средства “Каталог компьютеров” было разработано техническое задание, представленное в приложении А (см. стр. 23).

2. Описание блок-схем алгоритмов.

Схемы алгоритмов представлены в приложении Б (см. стр. 33). Ниже представлены описания алгоритмов.

2.1. Основная схема программы (рис. 1)

Данная схема декомпозирует основной блок программы, который включает себя приветствие пользователя, выделение памяти под списки, и вход в цикл, который представляет собой набор функций, которые выполняются при вводе соответствующего кода функции. Выход из цикла происходит, когда вводится код функции, которая предполагает выход из программы. Как только цикл был покинут, программа очищает память, ранее выделенную под списки, и прощается с пользователем.

2.2. Чтение данных из файла (рис. 2)

В данный алгоритм передаются списки, а также параметр, который указывает, были ли данные уже прочтены из файла. Так же алгоритм проверяет, есть ли в списке на момент выполнения функции. Если данные не были прочтены или пользователь желает перезаписать имеющиеся данные, то далее вводится директория к папке, которая содержит типизированные файлы с данными, затем вводится имя папки. При отсутствии в папке необходимых файлов выводится соответствующее сообщение, и управление передаётся обратно, иначе - данные прочтены и алгоритм завершает работу.

2.3. Выбор списка (рис. 3)

Функция возвращает номер выбранного списка.

2.4. Просмотр списков (рис. 4)

Пользователь вводит номер списка, который далее выводиться на экран.

2.5. Сортировка списков (рис. 5)

Пользователь вводит номер списка, в процедуру передаётся список, далее вводится номер поля, по которому идёт сортировка. Если список пуст или содержит 1 элемент, то сортировка не производится. Алгоритм сортировки заключается в поиске минимального элемента списка в промежутке от границы до конца списка и вставке его после границы, затем граница двигается на 1 элемент вправо. Алгоритм заканчивается, когда граница достигает конца списка.

Для списка комплектующих доступны следующие поля:

1) Код комплектующего
2) Код типа комплектующего
3) Производитель
4) Имя модели
5) Цена
6) Количество

Для списка типов комплектующих доступны следующие поля:
1) Код типа комплектующего
2) Название

Для списка совместимых комплектующих доступны следующие поля:
1) Код первого комплектующего
2) Код второго комплектующего

2.6. Поиск в списке (рис. 6)

Пользователь вводит номер списка, в процедуру передаётся список, вводится номер поля, по которому идёт поиск. Алгоритм проходиться по всему списку и, найдя элементы, выводит их, иначе - сообщение о то, что записи не найдены.

Для списка комплектующих доступны следующие поля:
1) Код комплектующего
2) Код типа комплектующего
3) Производитель
4) Имя модели

Для списка типов комплектующих доступны следующие поля:
1) Код типа комплектующего
2) Название

2.7. Добавление в список (рис. 7)

Пользователь вводит номер списка, далее:

- Для списка комплектующих передаются списки комп. и типов комп., выводится список типов комплектующих, чтобы пользователь мог выбрать нужный тип, выделяется память под новую запись, затем вводится тип комплектующего(если введён несуществующий, то алгоритм попросит ввести снова), производитель, имя модели(если будет введена уже имеющаяся модель, то алгоритм попросит ввести снова), параметры, цена и количество(если цена или количество будет меньше нуля, то алгоритм попросит ввести снова).

- Для списка типов комплектующих передаётся список типов комп., выделяется память под новую запись, вводится название(если такой тип уже есть, то алгоритм попросит ввести снова).
- Для списка совместимых комплектующих передаются списки комп. и совм. комп., выводится список комплектующих, чтобы пользователь выбрал нужные, выделяется память под новую запись, далее вводится код первого и второго комплектующего(если введённая запись уже существует, либо 1 или 2 коды введены некорректно, то алгоритм попросит ввести снова).

2.8. Удаление из списков (рис. 8)

Пользователь вводит номер списка, далее:

- Для списка комплектующих передаются списки комп. и совм. комп., выводится список комплектующих, чтобы пользователь выбрал нужные записи для удаления, далее вводится код комплектующего(если введён несуществующий код, то алгоритм попросит ввести снова), запись удаляется, память очищается. При этом происходит поиск удалённого комплектующего в списке совм. комп., и ,если таковые найдены, то в данном списке удаляются записи и очищается память.
- Для списка типов комплектующих передаются списки комп., типов комп. и совм. комп., выводится список типов комплектующих, чтобы пользователь выбрал нужные записи для удаления, далее вводиться код типа комплектующего(если введён несуществующий код, то алгоритм попросит ввести снова), запись удаляется, память очищается. При этом идёт поиск всех комплектующих в списке комплектующих, которые имеют удалённый тип, и, если такие найдены, записи удаляются, память очищается, и по алгоритму для списка комплектующих идёт поиск и удаление(если найдены) записей в списке совместимых комплектующих.
- Для списка совместимых комплектующих передаётся данный список, далее список выводится на экран, чтобы пользователь мог выбрать нужные записи для удаления, далее вводятся коды записи(если введена несуществующая запись, то алгоритм попросит ввести снова), запись удаляется, память очищается.

2.9. Редактирование в списках (рис. 9)

Пользователь вводит номер списка, далее в процедуру передаются списки(редактируемый и связные с ним), на экран выводится список для выбора записи для редактирования, далее вводится код записи(если введён несуществующий, то алгоритм попросит ввести снова). Далее пользователь редактирует поля(чтобы оставить поле неизменным, нужно нажать enter), проверка на ввод осуществляется так же, как и при добавлении элемента.

2.10. Специальные функции

2.10.1. Меню спецфункций (рис 10.1)

Функция возвращает код выбранной спецфункции.

2.10.2. Подбор всех возможных комбинаций компьютера в заданном ценовом диапазоне (рис 10.2)

2.10.2.1. Создание массива совместимых комплектующих (рис 10.2.1)

В функцию передаётся список совместимых комплектующих, далее внутри идёт проверка на совместимость всех имеющихся комплектующих друг с другом. Функция возвращает массив комплектующих, которые совместимы друг с другом и дают возможность собрать компьютер.

2.10.2.2 Получение всех комбинаций индексов (рис. 10.2.2)

В функцию передаётся длина массива совместимых комплектующих. Алгоритм находит все возможные сочетания без повторений. Функция возвращает массивы, которые представляют собой комбинации индексов.

2.10.2.3 Сортировка комбинаций (рис. 10.2.3)

В функцию передаётся массив с комбинациями и список комплектующих. Алгоритм сортирует все подобранные комбинации, а так же проверяет возможность приобретения той или иной комбинации(зависит, есть ли в наличии данные комплектующие).

В начале алгоритма проверяется, не были ли до этого подобраны комбинации, и, если были, то пользователю предлагается перезаписать, или оставить как есть. Далее пользователь вводит ценовой диапазон(если введено меньше нуля, то алгоритм попросит ввести снова). Если комбинации не были подобраны, выводится соответствующее сообщение, иначе - алгоритм возвращает массив всех возможных комбинаций сборки.

2.10.3. Оформление заказа (рис. 10.3)

В процедуру передаётся массив комбинаций. Далее на экран выводятся все возможные варианты для заказа, где пользователь решает, хочет ли он что-либо заказать или нет. Если да, то вводится директория, и создаётся текстовый файл с заказом.

2.10.4. Просмотр подобранных вариантов (рис. 10.4)

В процедуру передаётся массив комбинаций. Далее они выводятся на экран. Если пользователь желает сохранить в файл, то в начале вводит директорию, и данные сохраняются в файл.

2.11. Выход из программы

2.11.1 Выход без сохранения (рис 11.1)

Функция удаляет записи из списков и очищает память, результатом является код выхода из программы.

2.11.2 Выход с сохранением (рис 11.2)

Пользователь вводит директорию и имя папки, далее данные сохраняются в файл, удаляются записи из списков и очищается память. Результатом является код выхода из программы.

3. Выделение основных структур данных

Таблица 1 – Основные струкуры данных

1.Имя идентификатора структуры	2.Назначение структуры	3.Тип структуры
TCompPartsArr	Массив совместимых комплектующих	array of integer
TcompPartsMtx	Массив комбинаций сборки	array of TCompPartsArr
TcompPartsArrPrice	Массив стоимости комбинаций с соот. индексом	array of real
TCompPartsArrAvaliable	Массив с данными о возможности покупки комбинации с соот. индексом	array of boolean
TCombsFile	Текстовый файл для оформления заказа или записи комб. в файл	TextFile
PartListDataType	Структура списка комплектующих	packed record partCode: integer; partTypeCode: integer; manufacturer: TString; modelName: TString; parameters: TString; price: real; availability: integer; end;
PartListType	Список комплектующих	^PartListPointer
PartListPointer	Указатель на запись списка комплектующих	packed record lastID: integer; partListInfo: PartListDataType; partListNextElement: PartListType; end;

1	2	3
PartListFileType	Файл со структурой списка комплектующих	file of PartListDataType
PartTypeListDataType	Структура списка типов комплектующих	packed record partTypeCode: integer; partTypeName: TString; end;
PartTypeListType	Список типов комплектующих	^PartTypeListPointer
PartTypeListPointer	Указатель на запись списка типов комплектующих	packed record lastID: integer; partTypeListInfo: PartTypeListDataType; partTypeListNextElement: PartTypeListType; end;
PartTypeListFileType	Файл со структурой списка типов комплектующих	file of PartTypeListDataType
CompatiblePartListDataType	Структура списка совместимых комплектующих	packed record firstPartCode: integer; secondPartCode: integer; end;
CompatiblePartListType	Список совместимых комплектующих	^CompatiblePartListPointer
CompatiblePartListPointer	Указатель на запись списка совместимых комплектующих	packed record compatiblePartListInfo: CompatiblePartListDataType; compatiblePartListNextElement: CompatiblePartListType; end;
CompatiblePartListFileType	Файл со структурой списка типов комплектующих	file of CompatiblePartListDataType

4. Описание структур в виде таблиц

4.1. Список комплектующих

Таблица 2 – пример списка комплектующих

Код комп.	Код типа комп.	Изгото-витель	Имя модели	Пара-метры	Цена	Кол-во
1	1	Intel	Core i9 14900H	Frec 4.9 Ghz	799	1
2	2	Asus	B550HM	Socket lga 1700	599	2
3	3	HyperX	Fury	Frec 5200 Mhz	300	3
4	4	Nvidia	RTX 4090	8 Gb VRAM	1200	4

4.2. Список типов комрлектующих

Таблица 3 – пример списка типов комплектующих

Код типа комплектующего	Название
1	Central Processing Unit
2	MotherBoard
3	Random access memory
4	Graphical Processing Unit

4.3. Список совместимых комплектующих

Таблица 4 – пример списка совместимых комплектующих

Код первого комплектующего	Код второго комплектующего
1	2
1	3
1	4

4.4. Массив совместимых комплектующих

Хранит в себе все совместимые друг с другом комплектующие.

4.5. Массив комбинаций

Хранит все подобранные комбинации.

4.6. Массив сумм комбинаций

Хранит суммы подобранных комбинаций.

4.7. Массив с информацией о наличии

Хранит информацию о возможности приобретения.

5. Определение подпрограмм и их описание

Таблица 5 – Описание подрограмм

1.Имя подпрограммы	2.Назначение подпрограммы	3.Заголовок подпрограммы	4.Имя параметра	5.Назначение параметра
ClearScreen	Очистка консоли	procedure ClearScreen();	-	-
ReadFromFiles	Чтение данных из типизированных файлов	procedure ReadFromFiles(list1: PartListType; list2: PartTypeListType; list3: CompatiblePartListType; var isReadFromFile: boolean);	List1	Список комплектующих
			List2	Список типов комплектующих
			List3	Список совместимых комплектующих
			isReadFromFile	Пометка, были ли данные уже прочт.
GetList	Возвращает номер выбранного списка	function ShowListMenu(): integer; function SortListMenu(): integer; function FindInListMenu(): integer; function AddToListMenu(): integer; function DeleteFromListMenu(): integer; function EditInListMenu(): integer;	-	-

1	2	3	4	5
ShowList	Вывод списка на экран	procedure ShowPartList(list: PartListType); procedure ShowPartTypeList(list: PartTypeListType); procedure ShowCompatiblePartList(list: CompatiblePartListType);	List	Список, который выводится
SortList	Сортировка списка	procedure SortPartList(list: PartListType); procedure SortPartTypeList(list: PartTypeListType); procedure SortCompatiblePartList(list: CompatiblePartListType);	List	Список, который сортируется
FindInList	Поиск в списке	procedure FindInPartList(list: PartListType); procedure FindInPartTypeList(list: PartTypeListType); procedure FindInCompatiblePartList(list: CompatiblePartListType)	List	Список, в котором происходит поиск
AddToPartList	Добавление в список комплектующих	procedure AddToPartList(list: PartListType; checkList: PartTypeListType);	List	Список комплектующих
			CheckList	Список типов комплектующих

1	2	3	4	5
AddToPartTypeList	Добавление в список типов комплектующих	procedure AddToPartTypeList(list: PartTypeListType);	List	Список типов комплектующих
AddToCompatiblePartList	Добавление в список совмести-мых комплектующих	procedure AddToCompatiblePartList(list: CompatiblePartListType; checkList: PartListType);	List	Список совмести-мых комплектующих
			CheckList	Список комплектующих
DeleteFromPartList	Удаление из списка комплектующих	procedure DeleteFromPartList(list: PartListType; deleteList1: CompatiblePartListType);	List	Список комплектующих
			DeleteList1	Список типов комплектующих
DeleteFromPartTypeList	Удаление из списка типов комплектующих	procedure DeleteFromPartTypeList(deleteList1: PartListType; list: PartTypeListType; deleteList2: CompatiblePartListType);	List	Список типов комплектующих
			DeleteList1	Список комплектующих
			DeleteList2	Список совмести-мых комплектующих
DeleteFromCompatiblePartList	Удаление из списка совмести-мых комплектующих	procedure DeleteFromCompatiblePartList(list: CompatiblePartListType);	List	Список совмести-мых комплектующих

1	2	3	4	5
EditInList	Редактирование в списках	procedure EditInPartList(list: PartListType; checkList: PartTypeListType); procedure EditInPartTypeList(list: PartTypeListType); procedure EditInCompatiblePartList(list: CompatiblePartListType; checkList: PartListType);	List	Список для редактирования
SpecFunsMenu	Меню для выбора и исполнения спецфункций	function SpecialFunctionsMenu(isSpecFunCompleted: boolean): integer;	isSpecFunCompleted	Была ли спецфункция уже выполнена
			isNeededToUpdate	Указывает на валидность комбинаций
GetCompPrtsArr	Функция возвращает массив совместимых комплектующих	function GetCompatiblePartsArray(list: CompatiblePartListType): TCompPartsArr;	List	Список совмести-мых комплектующих
GetIndexArrCombs	Создание массива всевозможных комбинаций сборки	procedure GetAllCombsIndex(var IndexArr: TcompPartsMtx; n, m: integer);	IndexArr	Массив комбинаций индексов
			N	Значение N в сочетаниях
			m	Значение m в сочетаниях

1	2	3	4	5
SortCombs	Сортирует все комбинации и обозначает те, которые можно заказать	function SortCombs(list: PartListType; var IndexMtx: TcompPartsMtx; var avaliable: TCompPartsArrAvaliable): TcompPartsArrPrice;	List	Список комплектующих
			avaliable	Массив с информацией о возможности заказа комбинации
			IndexMtx	Массив комбинаций
ShowCombs	Вывод на экран комбинаций и сохранение данных в текстовый файл	function ShowAllCombs(list: PartListType; mtx: TcompPartsMtx; sum: TcompPartsArrPrice; price: real): integer;	List	Список комплектующих
			Mtx	Массив комбинаций
			Sum	Массив стоимостей комбинаций
			Price	Ценовой диапазон

1	2	3	4	5
MakeOrder	Запись заказа в текстовый файл	procedure MakeOrder(list: PartListType; mtx: TcompPartsMtx; sum: TcompPartsArrPrice; avaliable: TCompPartsArrAvaliable; size: integer; price: real);	List	Список комплектующих
			Mtx	Массив комбинаций
			Sum	Массив стоимостей комбинаций
			avaliable	Массив с инфор. о возм. покупки
			size	Кол-во комбинаций
			Price	Ценовой диапазон
			isNeededToUpdate	Указывает на валидность комбинаций

1	2	3	4	5
ExitFuns	Выход из программы(без сохранения, либо с сохранением в файл)	function SaveWithoutChanges(list1: PartListType; list2: PartTypeListType; list3: CompatiblePartListType): boolean; function SaveWithChanges(list1: PartListType; list2: PartTypeListType; list3: CompatiblePartListType): boolean;	List1	Список комплектующих
			List2	Список типов комплектующих
			List3	Список совместимых комплектующих
MainMenu	Главное меню программы	function MainMenu(): integer;	-	-

6. Тестирование и отладка программы

Таблица 6 – Тестирование ПС

1.№	2.Условие теста	3.Ожидаемый результат	4.Результат теста
1	Запуск программы	Программа успешно запущена	Тест пройден
2	Повторное чтение данных из файла	Сообщение, что повторное чтение невозможно	Тест пройден
3	Чтение данных из файла при наличии элементов в списках	Сообщение о наличии записей в списках и выбор(перезапись или отказ от чтения)	Тест пройден
4	Чтение данных из файла при отсутствии файлов по введенной директории	Сообщение, что файлы не найдены	Тест пройден
5	Чтение данных из файла при наличии файлов по введенной директории	Чтение данных из файла прошло успешно	Тест пройден
6	Просмотр списка	Список выведен на экран	Тест пройден
7	Сортировка пустого списка или списка с 1 записью	Список уже отсортирован, алгоритм не отработал	Тест пройден
8	Сортировка списка длиной более 1 элемента	Список отсортирован по возрастанию	Тест пройден
9	Поиск несуществующей записи	Сообщение о ненахождении записей	Тест пройден
10	Поиск записей с одинаковыми данными в искомых полях	Вывод найденных записей на экран	Тест пройден
11	Добавление в список новых записей при корректном вводе	Запись добавлена в список	Тест пройден
12	Добавление уже имеющейся записи в списке	Сообщение о дублировании записей	Тест пройден
13	Добавление с некорректным вводом данных	Сообщение о некорректном вводе	Тест пройден
14	Удаление записей при корректном вводе	Удаление записи из списка	Тест пройден
15	Удаление связных записей	Удаление записи и связн. с ней записей в др. сп.	Тест пройден

1	2	3	4
16	Удаление при некорректном вводе	Запись для удаления не найдена	Тест пройден
17	Редактирование полей записи при корректном вводе	Запись отредактирована	Тест пройден
18	Редактирование, которое влечет создание существующей записи	Сообщение о дублировании записей	Тест пройден
19	Подбор вариантов сборки при отсутствии совместимых комплектующих	Сообщение, что комбинации не подобраны	Тест пройден
20	Подбор вариантов сборки при вводе ценового диапазона	Комбинации подобраны в соответствии с ценовым диапазоном	Тест пройден
21	Просмотр комбинаций	Вывод комбинаций на экран	Тест пройден
22	Оформление заказа при отсутствии комплектующих	Заказ не может быть собран	Тест пройден
23	Оформление заказа при возможности его совершить	Заказ собран, сообщение о дальнейшей нужде о переподборе комбинаций	Тест пройден
24	Сохранение без изменений	Удаление записей, очистка памяти, успешный выход из программы0	Тест пройден
25	Сохранение с изменениями	Сохранение данных в файл	Тест пройден

7. Руководство пользователя

7.1. Руководство по установке

Для работы с программным средством необходимо установить и запустить программу CompCatalog.exe(рис. 1).

рис. 1 – Иконка программы

7.2. Руководство по эксплуатации

ПС имеет 10 функций меню(рис. 2):

1) Чтение из файла
2) Просмотр списков
3) Сортировка списков
4) Поиск в списках
5) Добавление в список
6) Удаление из списка
7) Редактирование в списках
8) Подбор вариантов сборки компьютера в заданном ценовом диапазоне и возможность заказа.
9) Выход без сохранения
10) Выход с сохранением

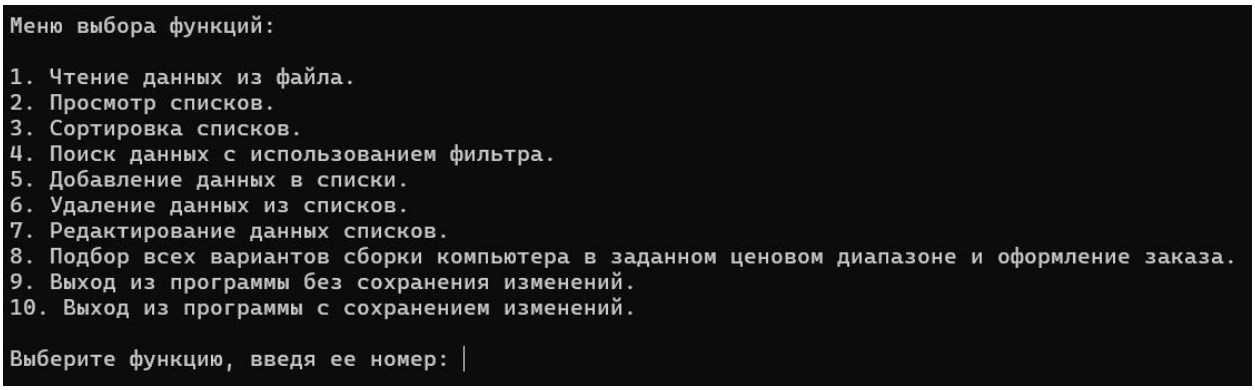


Рис. 2 – Меню программы

Функции 2, 3, 4, 5, 6, 7 предполагают взаимодействие со списками, поэтому для них будет появляться подменю списков(рис 3.):

1) Список комплектующих
2) Список типов комплектующих
3) Список совместимых комплектующих

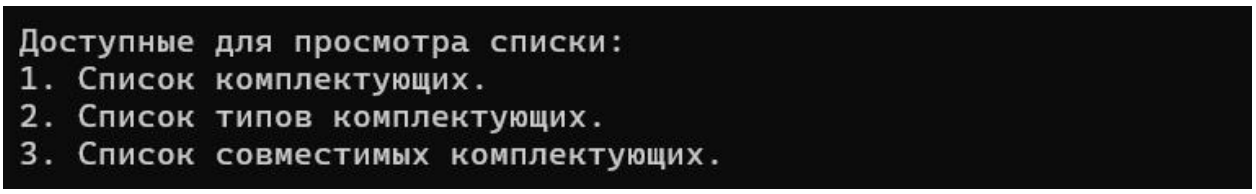


Рис. 3 – Подменю списков

При чтении из файла для корректной работы у пользователя должна быть директория и папка, которая включает в себя файлы с данными(рис. 4).

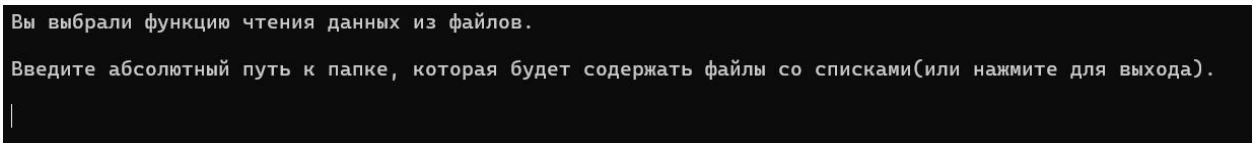


Рис. 4 – процедура чтения данных из файла

При просмотре пользователь выбирает список, которые далее выводится на экран(рис. 5).

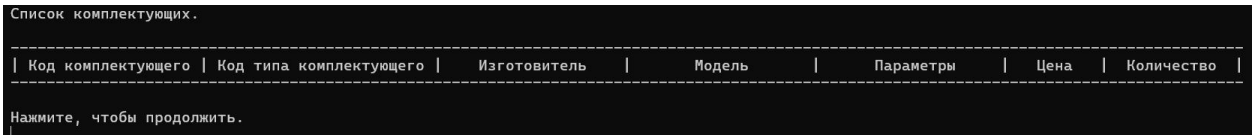


Рис. 5 – пример вывода списка

При сортировке пользователь выбирает список, далее для списка выбирает соот. фильтр, по которому идет сортировка по возрастанию(рис. 6).

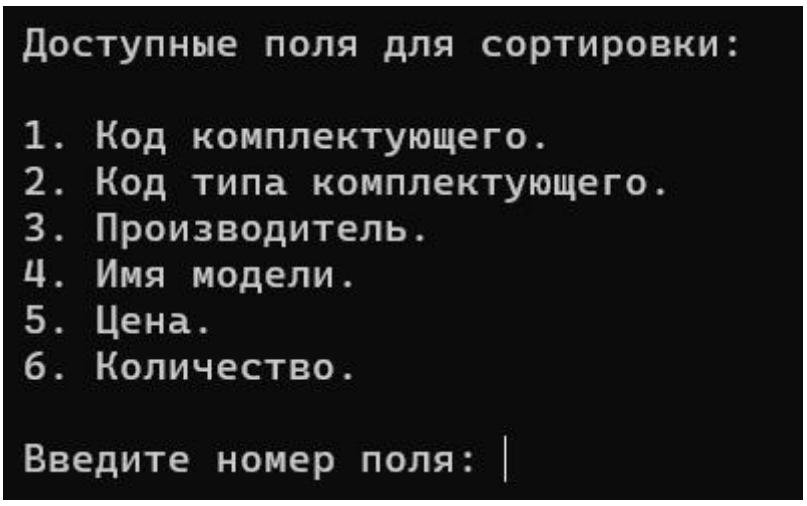


Рис. 6 – Поля, по которым возможна сортировка списка комплектующих

При поиске пользователь выбирает список, далее для списка выбирает соот. фильтр, по которому идет поиск записей(рис. 7).

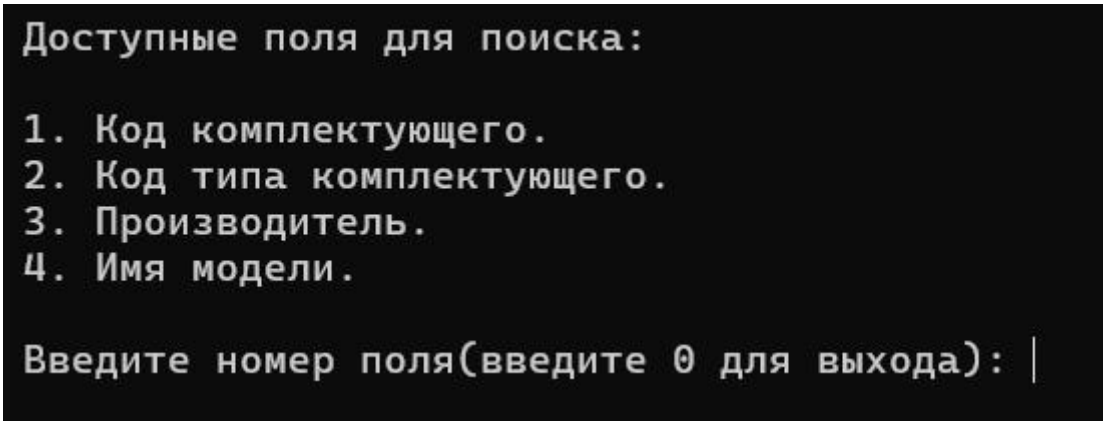


Рис. 7 – Поля, по которым возможен поиск в списке комплектующих

При добавлении записей пользователь выбирает список, далее заполняет необходимые поля. При некорректном вводе пользователя попросят ввести данные снова(рис. 8).

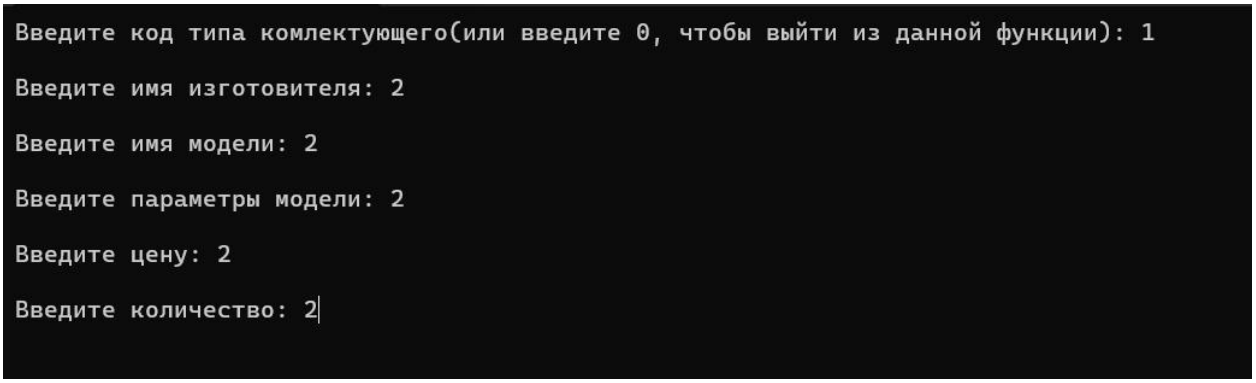


Рис. 8 – Пример добавления записи

При удалении записей пользователь выбирает список, далее происходит подтверждение удаления, и запись удаляется(рис. 9).

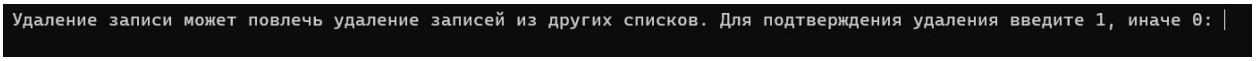


Рис. 9 – Подтверждение удаления

При редактировании записей пользователь выбирает список, далее выбирает запись для редактирования и меняет значения интересующих его полей(рис. 10).

```
Введите код типа комплектующего(нажмите для перехода к следующему полю или введите 0 для выхода):

Введите имя изготовителя(нажмите для перехода к следующему полю):

Введите имя модели(нажмите для перехода к следующему полю):

Введите параметры(нажмите для перехода к следующему полю): 21124

Введите цену(нажмите для перехода к следующему полю):

Введите количество(нажмите для перехода к следующему полю):
```

Рис. 10 – Пример редактирования записи

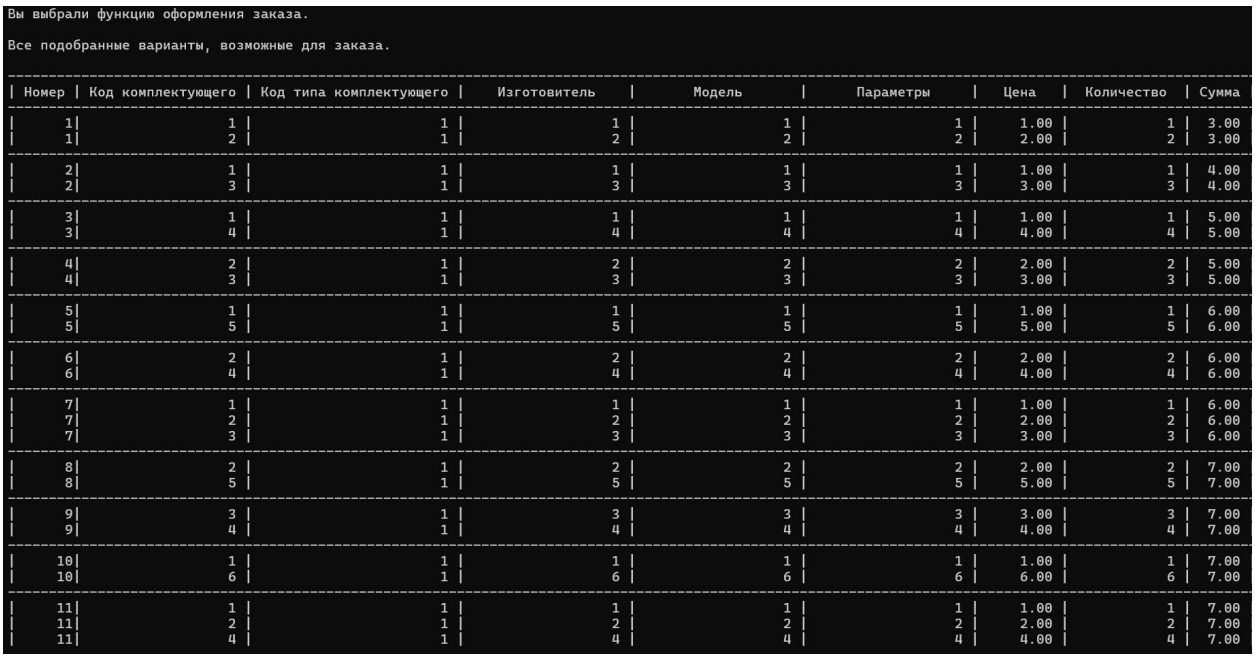
При подборе комплектующих пользователь вводит ценовой диапазон, и исходя из данных списка совместимых комплектующих и введенной цены подбираются всевозможные варианты сборки, доступные для заказа. После совершения покупки одного из заказов необходимо провести подбор вариантов заново для корректности данных(рис. 11, 12, 13).

```
Вы выбрали пункт специальных функций.

Доступные специальные функции:

1. Подбор всех возможных вариантов комплектации компьютера в заданном ценовом диапазоне.
2. Оформление заказа понравившегося варианта.
3. Просмотр подобранных комбинаций.

Выберите функцию, введя ее номер(0 для выхода):
```

Рис. 11 – подменю спецфункций

```
Все подобранные варианты.

| Номер | Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |      Параметры     |   Цена   | Количество | Сумма |
|     1|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  3.00 |
|     1|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  3.00 |
|     2|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  4.00 |
|     2|                  3 |                       1 |                  3 |                  3 |                  3 |     3.00 |          3 |  4.00 |
|     3|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  5.00 |
|     3|                  4 |                       1 |                  4 |                  4 |                  4 |     4.00 |          4 |  5.00 |
|     4|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  5.00 |
|     4|                  3 |                       1 |                  3 |                  3 |                  3 |     3.00 |          3 |  5.00 |
|     5|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  6.00 |
|     5|                  5 |                       1 |                  5 |                  5 |                  5 |     5.00 |          5 |  6.00 |
|     6|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  6.00 |
|     6|                  4 |                       1 |                  4 |                  4 |                  4 |     4.00 |          4 |  6.00 |
|     7|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  6.00 |
|     7|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  6.00 |
|     7|                  3 |                       1 |                  3 |                  3 |                  3 |     3.00 |          3 |  6.00 |
|     8|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  7.00 |
|     8|                  5 |                       1 |                  5 |                  5 |                  5 |     5.00 |          5 |  7.00 |
|     9|                  3 |                       1 |                  3 |                  3 |                  3 |     3.00 |          3 |  7.00 |
|     9|                  4 |                       1 |                  4 |                  4 |                  4 |     4.00 |          4 |  7.00 |
|    10|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  7.00 |
|    10|                  6 |                       1 |                  6 |                  6 |                  6 |     6.00 |          6 |  7.00 |
|    11|                  1 |                       1 |                  1 |                  1 |                  1 |     1.00 |          1 |  7.00 |
|    11|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  7.00 |
|    11|                  4 |                       1 |                  4 |                  4 |                  4 |     4.00 |          4 |  7.00 |
|    12|                  2 |                       1 |                  2 |                  2 |                  2 |     2.00 |          2 |  8.00 |
|    12|                  6 |                       1 |                  6 |                  6 |                  6 |     6.00 |          6 |  8.00 |
```

Рис. 12 – Просмотр комбинаций



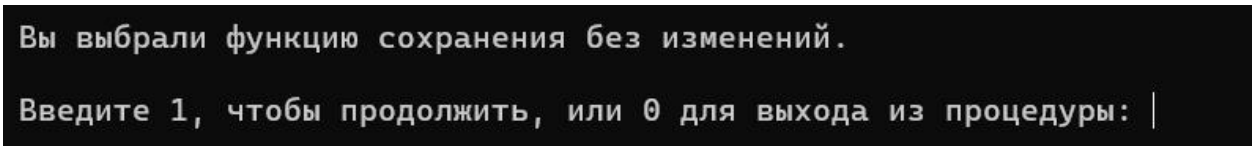
Вы выбрали функцию оформления заказа.

Все подобранные варианты, возможные для заказа.

Номер	Код комплектующего	Код типа комплектующего	Изготовитель	Модель	Параметры	Цена	Количество	Сумма
1	1	1	1	1	1	1.00	1	3.00
1	2	1	2	2	2	2.00	2	3.00
2	1	1	1	1	1	1.00	1	4.00
2	3	1	3	3	3	3.00	3	4.00
3	1	1	1	1	1	1.00	1	5.00
3	4	1	4	4	4	4.00	4	5.00
4	2	1	2	2	2	2.00	2	5.00
4	3	1	3	3	3	3.00	3	5.00
5	1	1	1	1	1	1.00	1	6.00
5	5	1	1	5	5	5.00	5	6.00
6	2	1	2	2	2	2.00	2	6.00
6	4	1	4	4	4	4.00	4	6.00
7	1	1	1	1	1	1.00	1	6.00
7	2	1	2	2	2	2.00	2	6.00
7	3	1	3	3	3	3.00	3	6.00
8	2	1	2	2	2	2.00	2	7.00
8	5	1	5	5	5	5.00	5	7.00
9	3	1	3	3	3	3.00	3	7.00
9	4	1	4	4	4	4.00	4	7.00
10	1	1	1	1	1	1.00	1	7.00
10	6	1	6	6	6	6.00	6	7.00
11	1	1	1	1	1	1.00	1	7.00
11	2	1	2	2	2	2.00	2	7.00
11	4	1	4	4	4	4.00	4	7.00

Рис. 13 – Оформление заказа

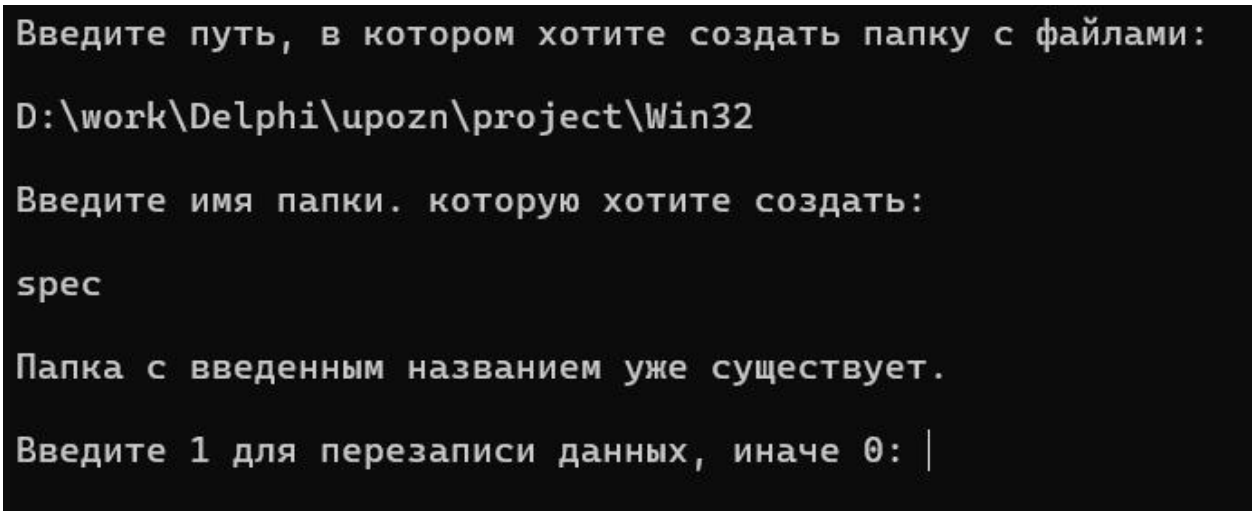
При сохранении без изменения пользователь должен подтвердить выход, чтобы случайно не потерять несохраненные данные(рис. 14).



```
Вы выбрали функцию сохранения без изменений.

Введите 1, чтобы продолжить, или 0 для выхода из процедуры:
```

Рис. 14 – Сохранение без изменений

При сохранении с изменениями пользователь вводит директорию и имя папки. При наличии введенной папки и файлов в ней, программа даст выбор: перезаписать или не изменять данные(рис. 15).



```
Введите путь, в котором хотите создать папку с файлами:

D:\work\Delphi\upozn\project\Win32

Введите имя папки. которую хотите создать:

spec

Папка с введенным названием уже существует.

Введите 1 для перезаписи данных, иначе 0:
```

Рис. 15 – Сохранение с изменениями

Приложение А

Техническое задание

Приложение Б

Схемы алгоритмов

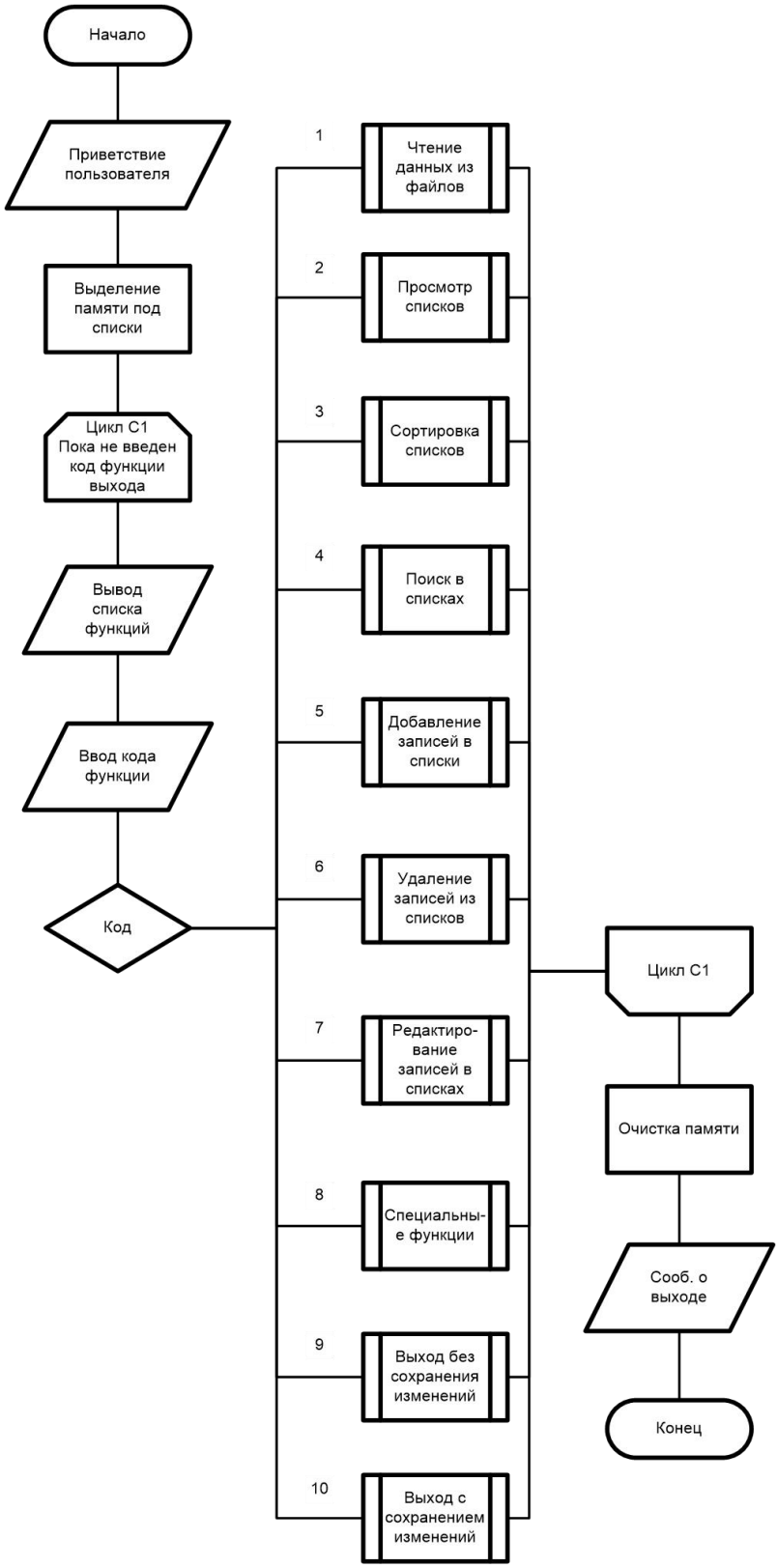


Рис. 1 - Основная схема программы

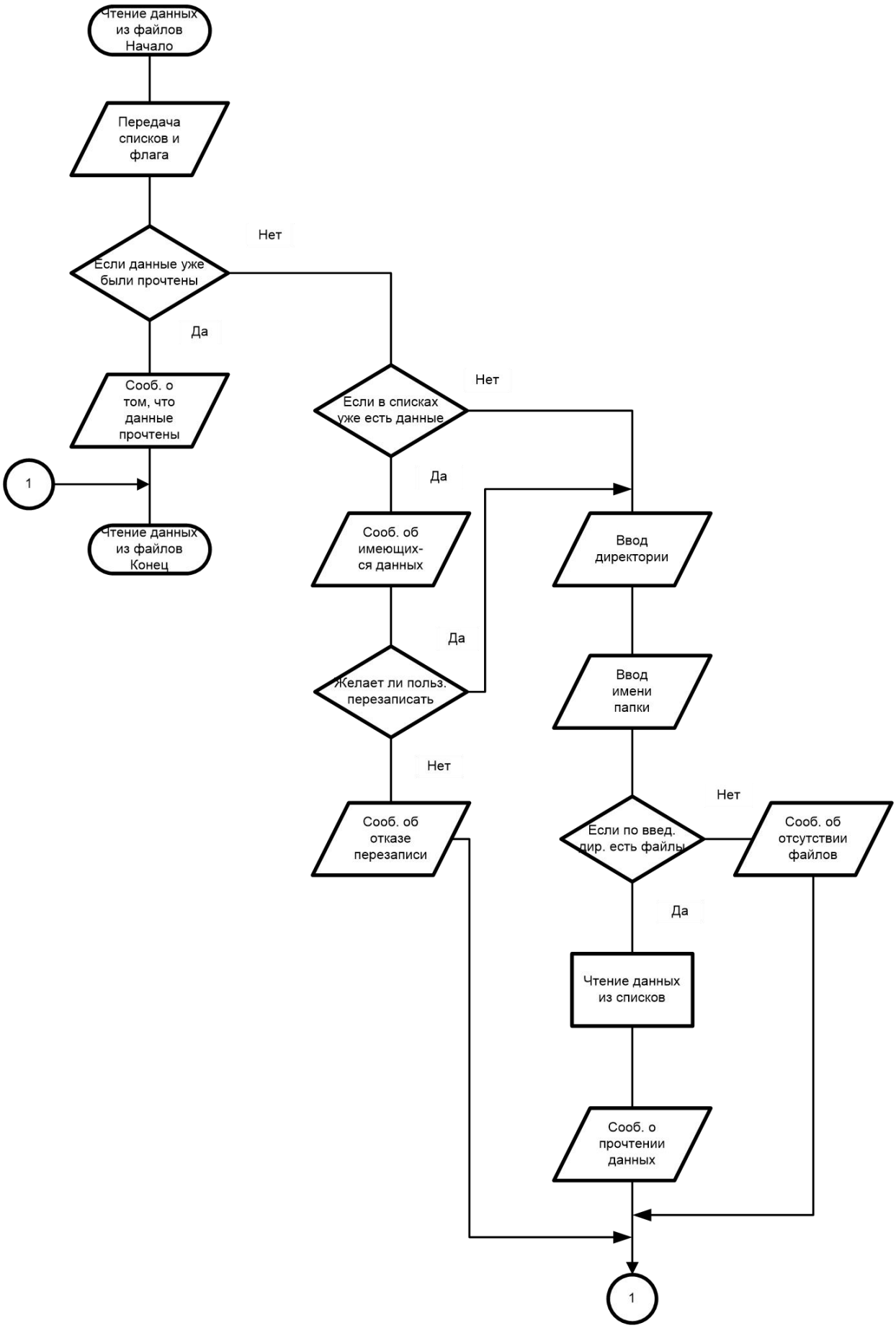


Рис. 2 - Чтение данных из файла

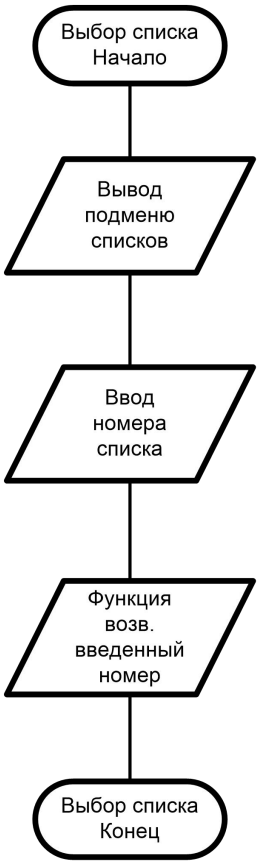


Рис. 3 - Выбор списка

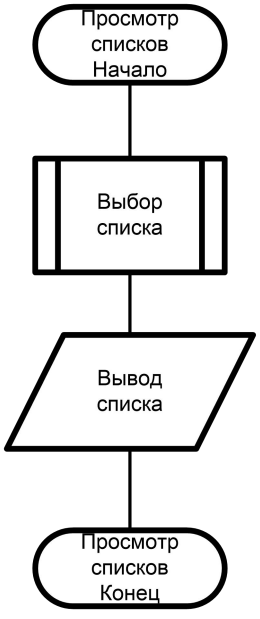


Рис. 4 - Просмотр списка

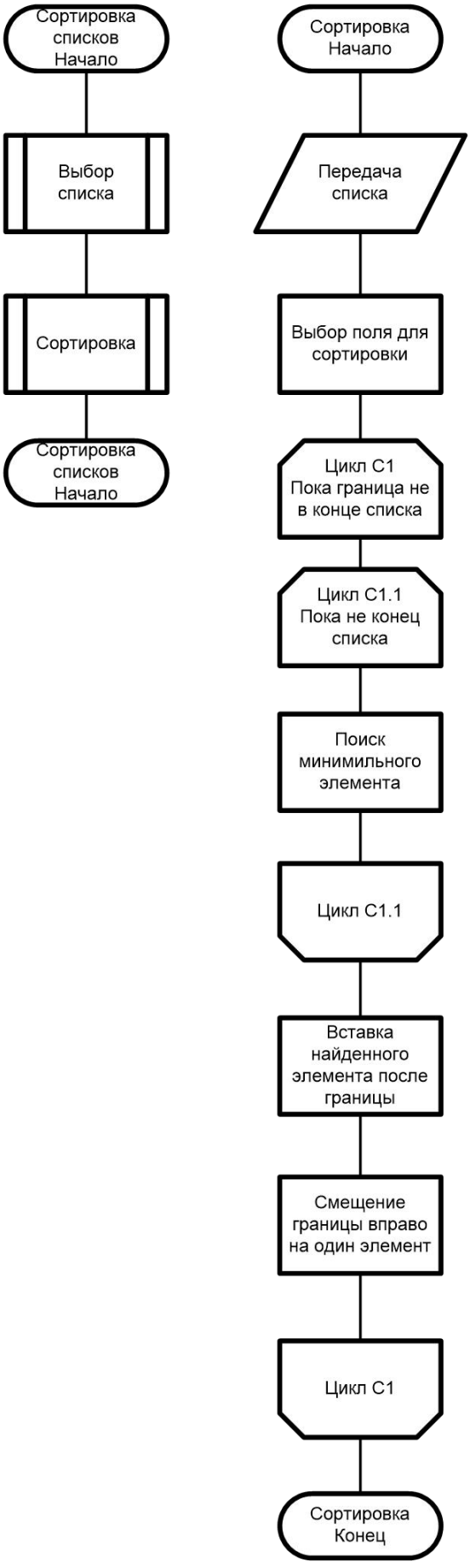


Рис. 5 - Сортировка списка

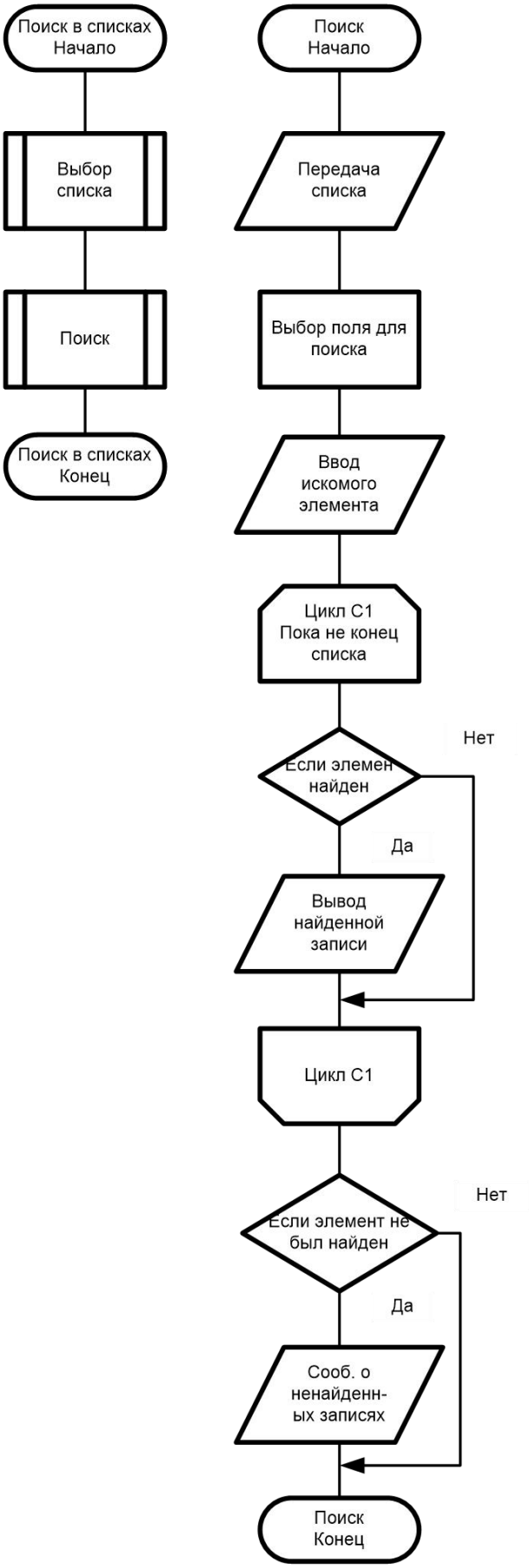


Рис. 6 - Поиск в списке

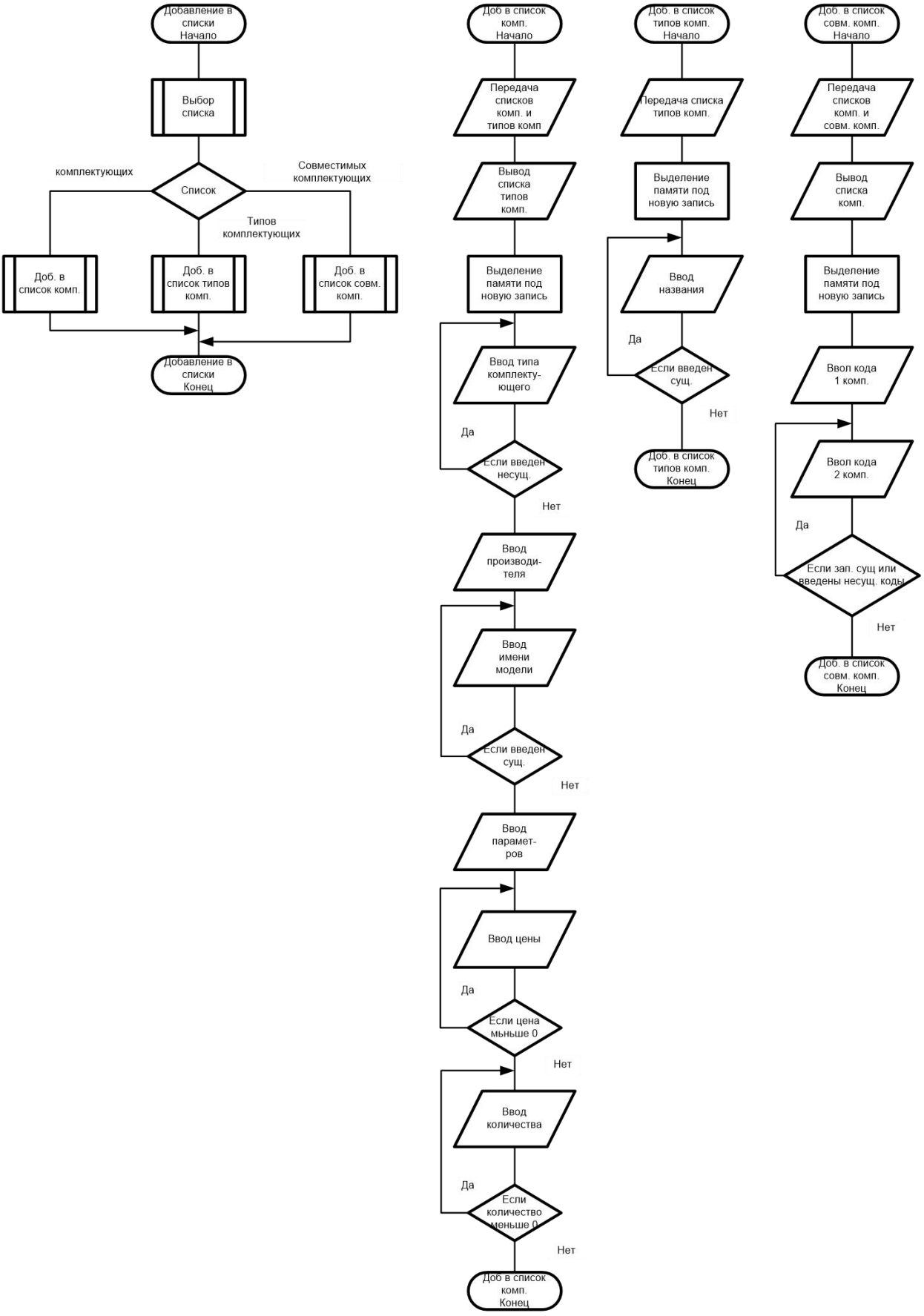


Рис. 7 - Добавление в списки

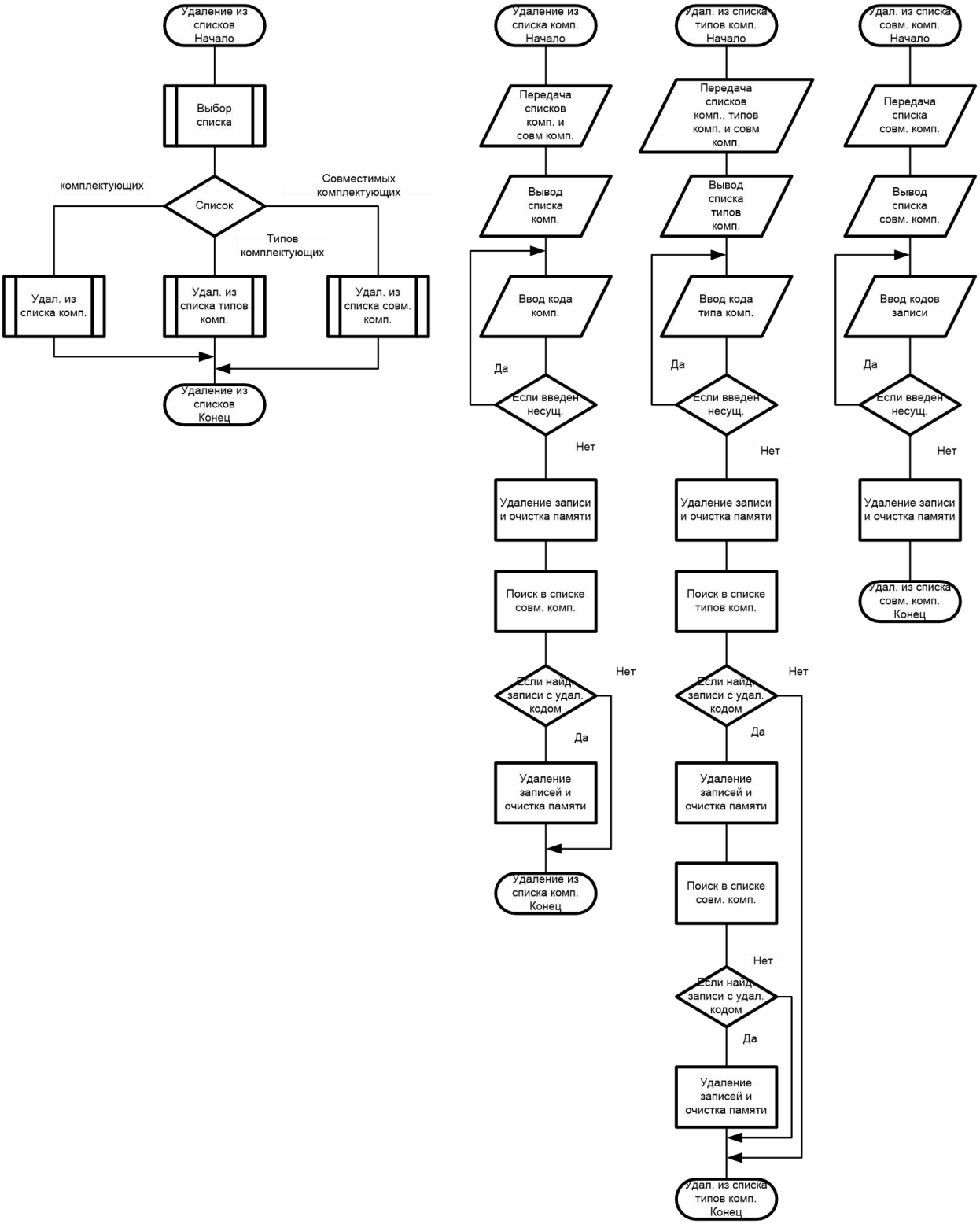


Рис. 8 - Удаление из списков

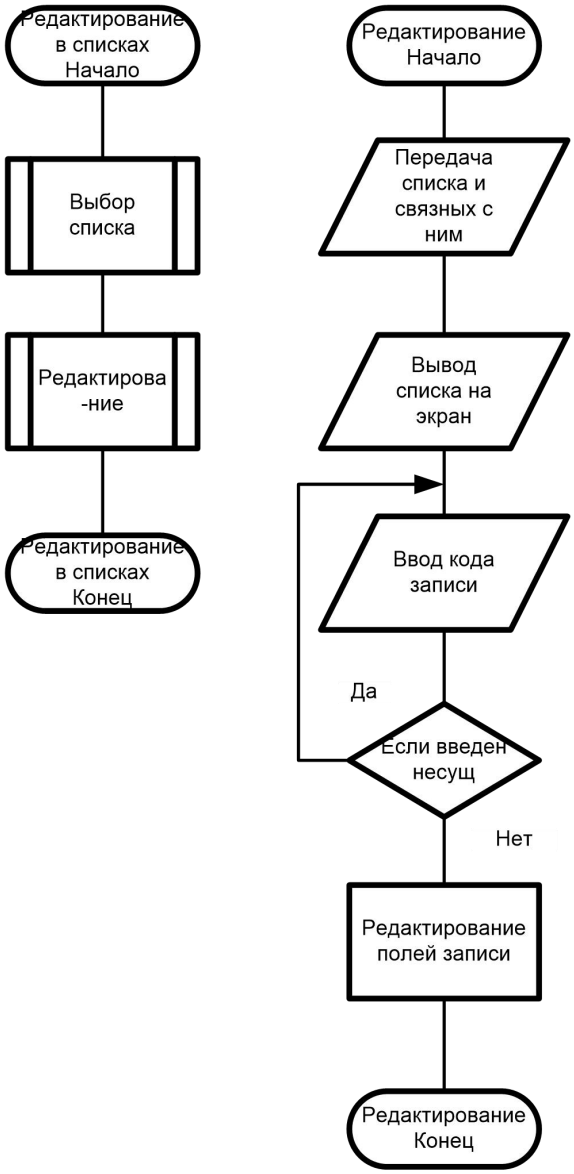


Рис. 9 - Редактирование в списках

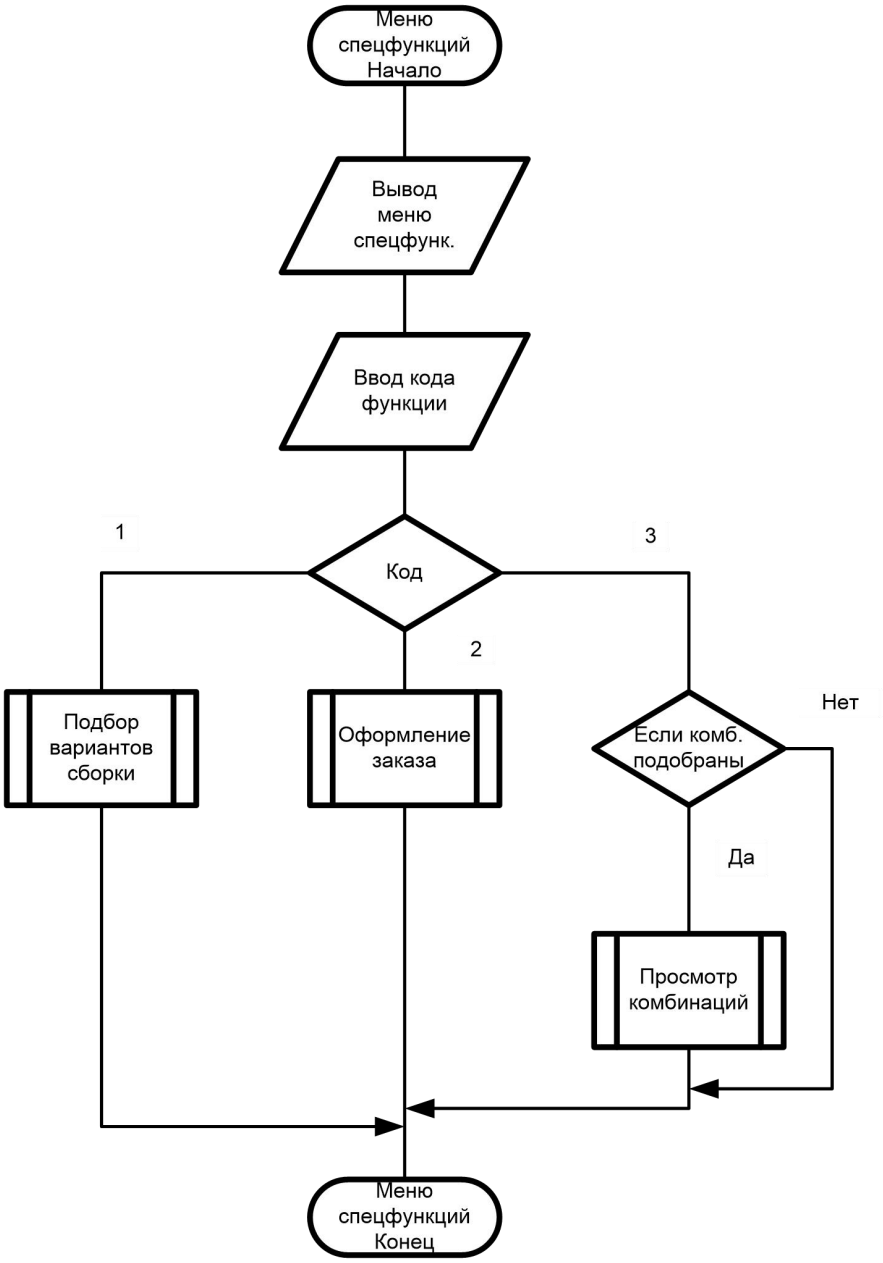


Рис. 10.1 - Меню спецфункций

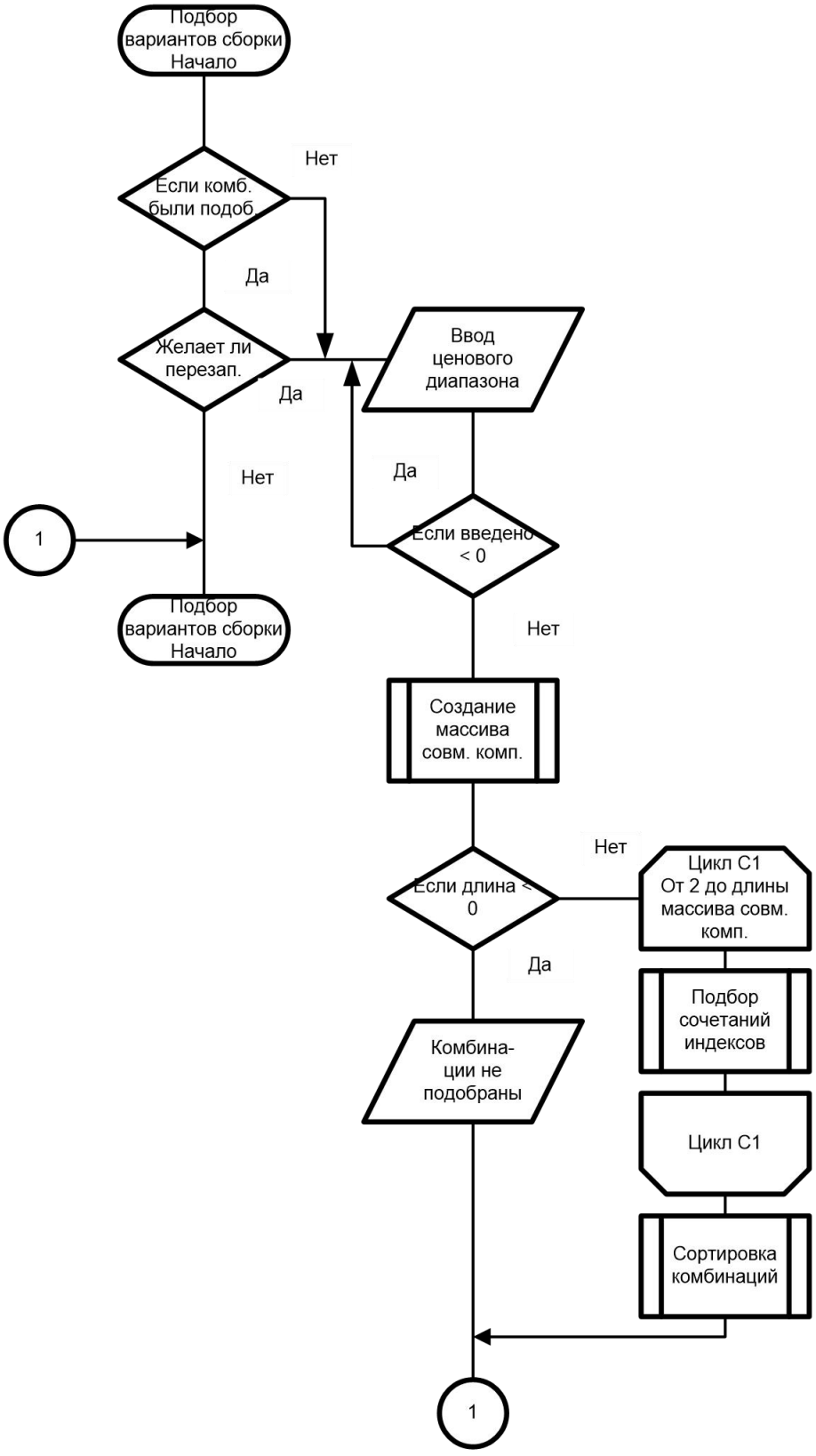


Рис 10.2 - Подбор всех возможных комбинаций компьютера в заданном ценовом диапазоне

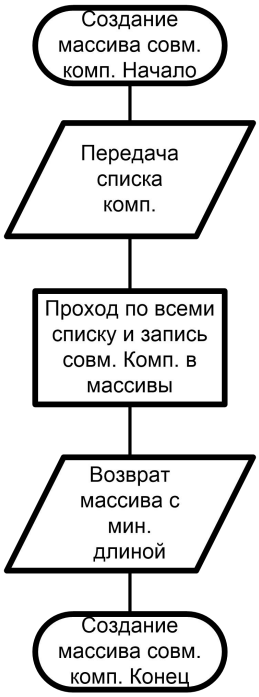


Рис. 10.2.1 - Создание массива совместимых комплектующих

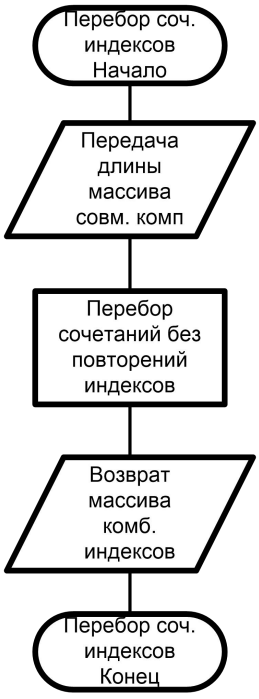


Рис. 10.2.2 - Перебор сочетаний индексов

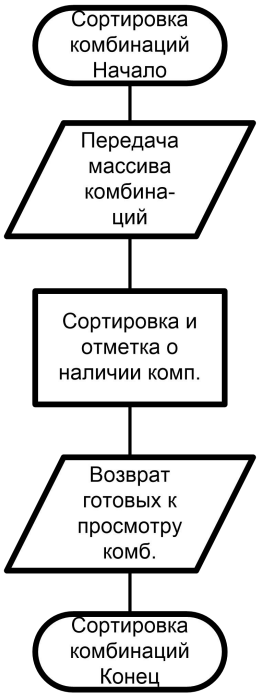


Рис 10.2.3 - Сортировка комбинаций

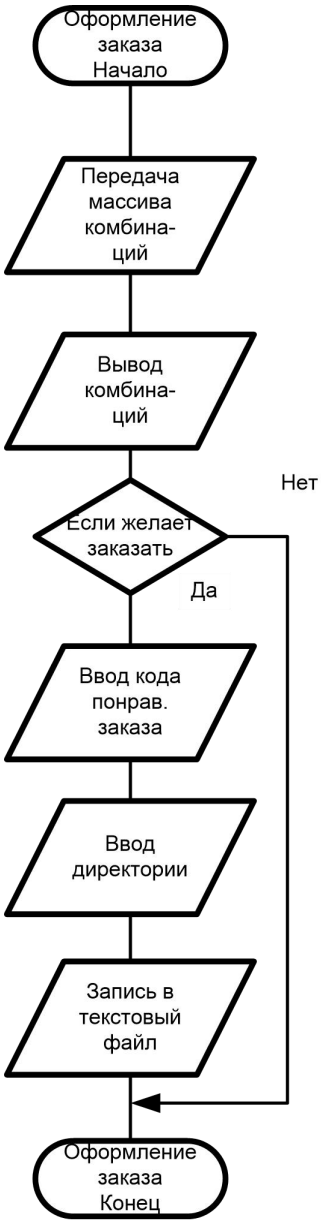


Рис 10.3 - Оформление заказа

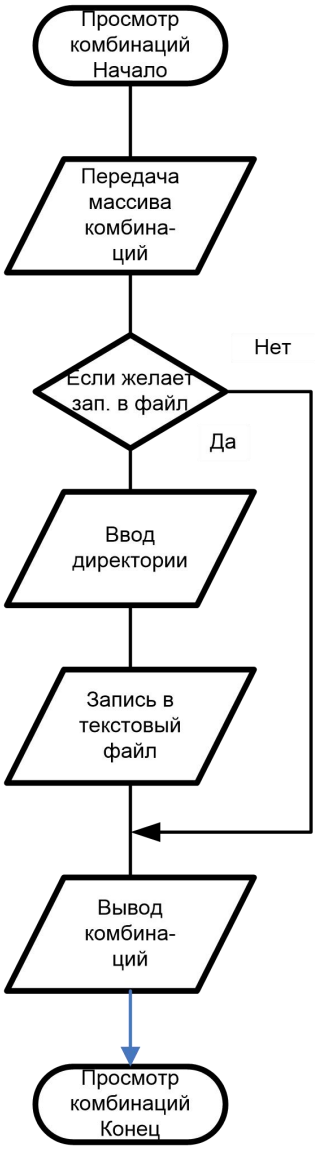


Рис 10.4 - Просмотр комбинаций

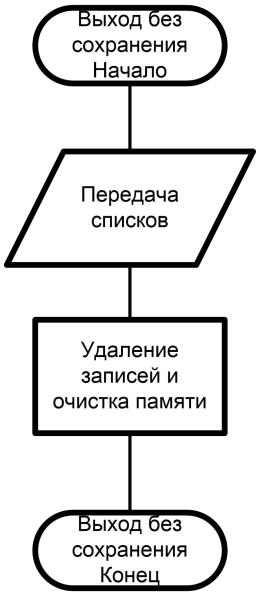


Рис 11.1 - Выход без сохранения

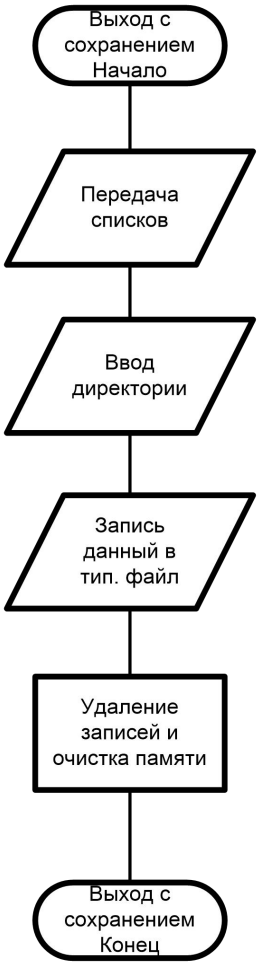


Рис 11.2 - Выход с сохранением

Приложение В

Основной код программы

```
program Project1;

{$APPTYPE CONSOLE}
{$R *.res}

{ used units and modules }
uses
  System.SysUtils, Windows;

{ types }
type
  TString = string[20];

  { specFuns data types }
  TCompPartsArr = array of integer;
  TcompPartsMtx = array of TCompPartsArr;
  TcompPartsArrPrice = array of real;
  TCompPartsArrAvaliable = array of boolean;
  TCombsFile = TextFile;

  { partList }
  PartListDataType = packed record
    partCode: integer;
    partTypeCode: integer;
    manufacturer: TString;
    modelName: TString;
    parameters: TString;
    price: real;
    availability: integer;
  end;

  PartListType = ^PartListPointer;

  PartListPointer = packed record
    lastID: integer;
    partListInfo: PartListDataType;
    partListNextElement: PartListType;
  end;

  PartListFileType = file of PartListDataType;

  { partTypeList }
  PartTypeListDataType = packed record
    partTypeCode: integer;
    partTypeName: TString;
  end;

  PartTypeListType = ^PartTypeListPointer;

  PartTypeListPointer = packed record
    lastID: integer;
    partTypeListInfo: PartTypeListDataType;
    partTypeListNextElement: PartTypeListType;
  end;

  PartTypeListFileType = file of PartTypeListDataType;

  { compatiblePartList }
```

```
  CompatiblePartListDataType = packed record
    firstPartCode: integer;
    secondPartCode: integer;
  end;

  CompatiblePartListType = ^CompatiblePartListPointer;

  CompatiblePartListPointer = packed record
    compatiblePartListInfo: CompatiblePartListDataType;
    compatiblePartListNextElement: CompatiblePartListType;
  end;

  CompatiblePartListFileType = file of CompatiblePartListDataType;
  { procedures }

  { console clear procedure }
procedure ClearScreen();
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));
  ConsoleSize := csbi.dwSize.X * csbi.dwSize.Y;
  Origin.X := 0;
  Origin.Y := 0;
  Win32Check(FillConsoleOutputCharacter(stdout, ' ', ConsoleSize, Origin,
    NumWritten));
  Win32Check(FillConsoleOutputAttribute(stdout, csbi.wAttributes, ConsoleSize,
    Origin, NumWritten));
  Win32Check(SetConsoleCursorPosition(stdout, Origin));
  sleep(115);
end;

{ procedure ReadFromFiles }
procedure ReadFromFiles(list1: PartListType; list2: PartTypeListType;
  list3: CompatiblePartListType; var isReadFromFile: boolean);

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

var
  header1: PartListType;
  header2: PartTypeListType;

var
  isAgreed: boolean;
  directoryPath, folder_files_name, path1, path2, path3: string;
  error1, error2, error3: integer;
  partListFile: PartListFileType;
  partTypeListFile: PartTypeListFileType;
  compatiblePartListFile: CompatiblePartListFileType;

begin
  isAgreed := false;
  header1 := list1;
  header2 := list2;
  ClearScreen();
```

```
  writeln('Вы выбрали функцию чтения данных из файлов.');
  writeln;
  if isReadFromFile then
  begin
    writeln('Данные из файлов уже были прочтены. Повторное чтение недоступно.');
    sleep(1200);
  end
  else
  begin
    if (list2^.partTypeListNextElement <> nil) then
    begin
      writeln('В списках уже имеются данные. Открытие файлов повлечет перезапись данных.');
      writeln;
      checkInput := '';
      checkErrorCode := 1;
      while checkErrorCode > 0 do
      begin
        write('Введите 1, чтобы прочитать данные из файлов, иначе введите 0: ');
        readln(checkInput);
        writeln;
        val(String(checkInput), checkInt, checkErrorCode);
        if (checkErrorCode > 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
          ClearScreen();
        end
        else
          case checkInt of
            1:
              isAgreed := true;
            0:
              isAgreed := false;
          else
            begin
              writeln('Некорректный ввод. Нажмите для повторного ввода.');
              readln;
              ClearScreen();
            end;
          end;
      end;
    end
    else
      isAgreed := true;
    if not isAgreed then
    begin
      ClearScreen();
      writeln('Вы отказались от открытия файлов.');
      sleep(1200);
    end
    else
    begin
      repeat
        writeln('Введите абсолютный путь к папке, которая будет содержать файлы со списками(или нажмите
для выхода).');
        writeln;
        readln(directoryPath);
        writeln;
        if (not directoryExists(directoryPath)) and (directoryPath <> '') then
        begin
          writeln('Такой директории не существует. Нажмите для повторного ввода.');
          readln;
          ClearScreen();
```

```
      end;
    until (directoryExists(directoryPath)) or (directoryPath = '');
    if directoryPath = '' then
    begin
      ClearScreen();
      writeln('Вы отказались от открытия файлов.');
      sleep(1200);
    end
    else
    begin
      repeat
        writeln('Введите имя папки(или нажмите для выхода): ');
        writeln;
        readln(folder_files_name);
        writeln;
        if (not directoryExists(directoryPath + '\' + folder_files_name)) and
          (folder_files_name <> '') then
        begin
          writeln('Такой папки не существует. Нажмите для повторного ввода.');
          readln;
          ClearScreen();
        end;
      until (directoryExists(directoryPath + '\' + folder_files_name)) or
        (folder_files_name = '');
      if folder_files_name = '' then
      begin
        ClearScreen();
        writeln('Вы отказались от открытия файлов.');
        sleep(1200);
      end
      else
      begin
        directoryPath := directoryPath + '\' + folder_files_name;
        path1 := directoryPath + '\' + folder_files_name +
          '_PartListData.upozn';
        path2 := directoryPath + '\' + folder_files_name +
          '_PartTypeListData.upozn';
        path3 := directoryPath + '\' + folder_files_name +
          '_CompatiblePartListData.upozn';
{$I-}
        assignFile(partListFile, path1);
        reset(partListFile);
        error1 := IOResult;
        assignFile(partTypeListFile, path2);
        reset(partTypeListFile);
        error2 := IOResult;
        assignFile(compatiblePartListFile, path3);
        reset(compatiblePartListFile);
        error3 := IOResult;
        if not((error1 = 0) and (error2 = 0) and (error3 = 0)) then
        begin
          writeln('В введенной директории отсутствуют файлы.');
          writeln;
          writeln('Нажмите, чтобы продолжить.');
          readln;
        end
        else
        begin
          while not EOF(partListFile) do
          begin
            new(list1^.partListNextElement);
            list1 := list1^.partListNextElement;
            read(partListFile, list1^.partListInfo);
```

```
          end;
          list1^.partListNextElement := nil;
          header1^.lastID := list1^.partListInfo.partCode;
          while not EOF(partTypeListFile) do
          begin
            new(list2^.partTypeListNextElement);
            list2 := list2^.partTypeListNextElement;
            read(partTypeListFile, list2^.partTypeListInfo);
          end;
          list2^.partTypeListNextElement := nil;
          header2^.lastID := list2^.partTypeListInfo.partTypeCode;
          while not EOF(compatiblePartListFile) do
          begin
            new(list3^.compatiblePartListNextElement);
            list3 := list3^.compatiblePartListNextElement;
            read(compatiblePartListFile, list3^.compatiblePartListInfo);
          end;
          list3^.compatiblePartListNextElement := nil;
          closeFile(partListFile);
          closeFile(partTypeListFile);
          closeFile(compatiblePartListFile);
          ClearScreen();
          writeln('Данные прочтены.');
          sleep(1200);
          isReadFromFile := true;
        end;
      end;
    end;
  end;
 end;
end;

{ ShowListFunctions }
{ function ShowListMenu }
function ShowListMenu(): integer;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3)) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию просмотра списков.');
    writeln;
    writeln('Доступные для просмотра списки: ');
    writeln('1. Список комплектующих.');
    writeln('2. Список типов комплектующих.');
    writeln('3. Список совместимых комплектующих.');
    writeln;
    write('Введите номер списка, который хотите просмотреть(или введите 0, чтобы выйти из функции
просмотра): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3)) then
    begin
      writeln('Списка с заданным номером не существует. Нажмите для повторного ввода.');
      readln;
```

```
    end;
  end;
  result := checkInt;
end;

{ procedure ShowPartList }
procedure ShowPartList(list: PartListType);

begin
  ClearScreen();
  writeln('Список комплектующих.');
  writeln;
  writeln('-------------------------------------------------------------------------------------------------------------------------------
---------');
  writeln('| Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |
Параметры    |   Цена   | Количество  |');
  writeln('-------------------------------------------------------------------------------------------------------------------------------
---------');
  list := list^.partListNextElement;
  while list <> nil do
  begin
    writeln('|', list^.partListInfo.partCode:19, ' |',
      list^.partListInfo.partTypeCode:24, ' |', list^.partListInfo.manufacturer
      :19, ' |', list^.partListInfo.modelName:19, ' |',
      list^.partListInfo.parameters:19, ' |', list^.partListInfo.price:9:2,
      ' |', list^.partListInfo.availability:13, ' |');
    writeln('-------------------------------------------------------------------------------------------------------------------------------
----------');
    list := list^.partListNextElement;
  end;
  writeln;
  writeln('Нажмите, чтобы продолжить.');
  readln;
end;

{ procedure ShowPartTypeList }
procedure ShowPartTypeList(list: PartTypeListType);

begin
  ClearScreen();
  writeln('Список типов комплектующих.');
  writeln;
  writeln('-------------------------------------------------');
  writeln('| Код типа комплектующего |      Название      |');
  writeln('-------------------------------------------------');
  list := list^.partTypeListNextElement;
  while list <> nil do
  begin
    writeln('|', list^.partTypeListInfo.partTypeCode:24, ' |',
      list^.partTypeListInfo.partTypeName:19, ' |');
    writeln('-------------------------------------------------');
    list := list^.partTypeListNextElement;
  end;
  writeln;
  writeln('Нажмите, чтобы продолжить.');
  readln;
end;

{ procedure ShowCompatiblePartList }
procedure ShowCompatiblePartList(list: CompatiblePartListType);

begin
  ClearScreen();
```

```
  writeln('Список совместимых комплектующих.');
  writeln;
  writeln('------------------------------------------------------------');
  writeln('| Код первого комплектующего | Код второго комплектующего |');
  writeln('------------------------------------------------------------');
  list := list^.compatiblePartListNextElement;
  while list <> nil do
  begin
    writeln('|', list^.compatiblePartListInfo.firstPartCode:27, ' |',
      list^.compatiblePartListInfo.secondPartCode:27, ' |');
    writeln('------------------------------------------------------------');
    list := list^.compatiblePartListNextElement;
  end;
  writeln;
  writeln('Нажмите, чтобы продолжить.');
  readln;
end;

{ SortListFunctions }
{ function SortListMenu }
function SortListMenu(): integer;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3)) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию сортировки списков.');
    writeln;
    writeln('Доступные для сортировки списки: ');
    writeln('1. Список комплектующих.');
    writeln('2. Список типов комплектующих.');
    writeln('3. Список совместимых комплектующих.');
    writeln;
    write('Введите номер списка, который хотите сортировать(или введите 0, чтобы выйти из функции
сортировки): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3)) then
    begin
      writeln('Списка с заданным номером не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  result := checkInt;
end;

{ procedure SortPartList }
procedure SortPartList(list: PartListType);

  function CurrPrevComp(code: integer; curr, prev: PartListType): boolean;

  var
    comp: boolean;

  begin
```

```
  comp := false;
  case code of
    1:
      comp := (curr^.partListInfo.partCode >=
        prev^.partListNextElement^.partListInfo.partCode);
    2:
      comp := (curr^.partListInfo.partTypeCode >=
        prev^.partListNextElement^.partListInfo.partTypeCode);
    3:
      comp := (curr^.partListInfo.manufacturer >=
        prev^.partListNextElement^.partListInfo.manufacturer);
    4:
      comp := (curr^.partListInfo.modelName >=
        prev^.partListNextElement^.partListInfo.modelName);
    5:
      comp := (curr^.partListInfo.price >=
        prev^.partListNextElement^.partListInfo.price);
    6:
      comp := (curr^.partListInfo.availability >=
        prev^.partListNextElement^.partListInfo.availability);
  end;
  result := comp;
end;

procedure SortPartListElements(list: PartListType; fieldCode: integer);

var
  sorted, curr, prev: PartListType;
  comparator1: boolean;

begin
  comparator1 := false;
  sorted := list;
  list := list^.partListNextElement;
  sorted^.partListNextElement := nil;
  while list <> nil do
  begin
    curr := list;
    list := list^.partListNextElement;
    case fieldCode of
      1:
        comparator1 := (curr^.partListInfo.partCode <
          sorted^.partListInfo.partCode);
      2:
        comparator1 := (curr^.partListInfo.partTypeCode <
          sorted^.partListInfo.partTypeCode);
      3:
        comparator1 := (curr^.partListInfo.manufacturer <
          sorted^.partListInfo.manufacturer);
      4:
        comparator1 := (curr^.partListInfo.modelName <
          sorted^.partListInfo.modelName);
      5:
        comparator1 :=
          (curr^.partListInfo.price < sorted^.partListInfo.price);
      6:
        comparator1 := (curr^.partListInfo.availability <
          sorted^.partListInfo.availability)
    end;
    if comparator1 then
    begin
      curr^.partListNextElement := sorted;
      sorted := curr;
```

```
      end
      else
      begin
        prev := sorted;
        while (prev^.partListNextElement <> nil) and
          (CurrPrevComp(fieldCode, curr, prev)) do
          prev := prev^.partListNextElement;
        curr^.partListNextElement := prev^.partListNextElement;
        prev^.partListNextElement := curr;
      end;
    end;
    // list := sorted;
  end;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 6)) do
  begin
    ClearScreen();
    writeln('Доступные поля для сортировки: ');
    writeln;
    writeln('1. Код комплектующего.');
    writeln('2. Код типа комплектующего.');
    writeln('3. Производитель.');
    writeln('4. Имя модели.');
    writeln('5. Цена.');
    writeln('6. Количество.');
    writeln;
    write('Введите номер поля: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 6)) then
    begin
      writeln('Введенное поле не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if not((list = nil) or (list^.partListNextElement = nil)) then
    SortPartListElements(list, checkInt);
  ClearScreen();
  writeln('Список отсортирован.');
  sleep(1200);
end;

{ procedure SortPartTypeList }
procedure SortPartTypeList(list: PartTypeListType);

  function CurrPrevComp(code: integer; curr, prev: PartTypeListType): boolean;

  var
    comp: boolean;

  begin
    comp := false;
    case code of
      1:
```

```
      comp := (curr^.partTypeListInfo.partTypeCode >=
        prev^.partTypeListNextElement^.partTypeListInfo.partTypeCode);
    2:
      comp := (curr^.partTypeListInfo.partTypeName >=
        prev^.partTypeListNextElement^.partTypeListInfo.partTypeName);
  end;
  result := comp;
end;

procedure SortPartTypeListElements(list: PartTypeListType;
  fieldCode: integer);

var
  sorted, curr, prev: PartTypeListType;
  comparator1: boolean;

begin
  comparator1 := false;
  sorted := list;
  list := list^.partTypeListNextElement;
  sorted^.partTypeListNextElement := nil;
  while list <> nil do
  begin
    curr := list;
    list := list^.partTypeListNextElement;
    case fieldCode of
      1:
        comparator1 := (curr^.partTypeListInfo.partTypeCode <
          sorted^.partTypeListInfo.partTypeCode);
      2:
        comparator1 := (curr^.partTypeListInfo.partTypeName <
          sorted^.partTypeListInfo.partTypeName);
    end;
    if comparator1 then
    begin
      curr^.partTypeListNextElement := sorted;
      sorted := curr;
    end
    else
    begin
      prev := sorted;
      while (prev^.partTypeListNextElement <> nil) and
        (CurrPrevComp(fieldCode, curr, prev)) do
        prev := prev^.partTypeListNextElement;
      curr^.partTypeListNextElement := prev^.partTypeListNextElement;
      prev^.partTypeListNextElement := curr;
    end;
  end;
  // list := sorted;
end;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 2)) do
  begin
    ClearScreen();
    writeln('Доступные поля для сортировки: ');
```

```
    writeln;
    writeln('1. Код типа комплектующего.');
    writeln('2. Название.');
    writeln;
    write('Введите номер поля: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 2)) then
    begin
      writeln('Введенное поле не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if not((list = nil) or (list^.partTypeListNextElement = nil)) then
    SortPartTypeListElements(list, checkInt);
  ClearScreen();
  writeln('Список отсортирован.');
  sleep(1200);
end;

{ procedure SortCompatiblePartList }
procedure SortCompatiblePartList(list: CompatiblePartListType);

  function CurrPrevComp(code: integer;
    curr, prev: CompatiblePartListType): boolean;

  var
    comp: boolean;

  begin
    comp := false;
    case code of
      1:
        comp := (curr^.compatiblePartListInfo.firstPartCode >=
          prev^.compatiblePartListNextElement^.compatiblePartListInfo.
          firstPartCode);
      2:
        comp := (curr^.compatiblePartListInfo.secondPartCode >=
          prev^.compatiblePartListNextElement^.compatiblePartListInfo.
          secondPartCode);
    end;
    result := comp;
  end;

  procedure SortCompatiblePartListElements(list: CompatiblePartListType;
    fieldCode: integer);

  var
    sorted, curr, prev: CompatiblePartListType;
    comparator1: boolean;

  begin
    comparator1 := false;
    sorted := list;
    list := list^.compatiblePartListNextElement;
    sorted^.compatiblePartListNextElement := nil;
    while list <> nil do
    begin
      curr := list;
      list := list^.compatiblePartListNextElement;
      case fieldCode of
        1:
```

```
        comparator1 := (curr^.compatiblePartListInfo.firstPartCode <
          sorted^.compatiblePartListInfo.firstPartCode);
      2:
        comparator1 := (curr^.compatiblePartListInfo.secondPartCode <
          sorted^.compatiblePartListInfo.secondPartCode);
    end;
    if comparator1 then
    begin
      curr^.compatiblePartListNextElement := sorted;
      sorted := curr;
    end
    else
    begin
      prev := sorted;
      while (prev^.compatiblePartListNextElement <> nil) and
        (CurrPrevComp(fieldCode, curr, prev)) do
        prev := prev^.compatiblePartListNextElement;
      curr^.compatiblePartListNextElement :=
        prev^.compatiblePartListNextElement;
      prev^.compatiblePartListNextElement := curr;
    end;
  end;
  // list := sorted;
 end;

var
 checkInput: TString;
 checkInt, checkErrorCode: integer;

begin
 checkInput := '';
 checkInt := 0;
 checkErrorCode := 1;
 while (checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 2)) do
 begin
  ClearScreen();
  writeln('Доступные поля для сортировки: ');
  writeln;
  writeln('1. Код первого комплектующего.');
  writeln('2. Код второго комплектующего.');
  writeln;
  write('Введите номер поля: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if (checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 2)) then
  begin
    writeln('Введенное поле не существует. Нажмите для повторного ввода.');
    readln;
  end;
 end;
 if not((list = nil) or (list^.compatiblePartListNextElement = nil)) then
  SortCompatiblePartListElements(list, checkInt);
 ClearScreen();
 writeln('Список отсортирован.');
 sleep(1200);
end;

{ FindInListFunctions }
{ function FindInListMenu }
function FindInListMenu(): integer;

var
```

```
  checkInput: TString;
  checkInt, checkErrorCode: integer;

begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3)) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию поиска данных в списках.');
    writeln;
    writeln('Доступные для поиска списки: ');
    writeln('1. Список комплектующих.');
    writeln('2. Список типов комплектующих.');
    writeln('3. Список совместимых комплектующих.');
    writeln;
    write('Введите номер списка,в котором хотите провести поиск(или введите 0, чтобы выйти из функции
поиска): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3)) then
    begin
      writeln('Списка с заданным номером не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  result := checkInt;
end;

{ procedure FindInPartList }
procedure FindInPartList(list: PartListType);

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

var
  fieldCode: integer;
  isTableShown, isNotExit, isNotExitCheck, comparator: boolean;
  header: PartListType;

begin
  header := list;
  isNotExit := true;
  comparator := false;
  while isNotExit do
  begin
    checkInput := '';
    checkInt := 0;
    checkErrorCode := 1;
    while (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 4)) do
    begin
      ClearScreen();
      writeln('Доступные поля для поиска: ');
      writeln;
      writeln('1. Код комплектующего.');
      writeln('2. Код типа комплектующего.');
      writeln('3. Производитель.');
      writeln('4. Имя модели.');
      writeln;
      write('Введите номер поля(введите 0 для выхода): ');
```

```
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 4)) then
  begin
    writeln('Введенное поле не существует. Нажмите для повторного ввода.');
    readln;
  end;
end;
if checkInt <> 0 then
begin
  fieldCode := checkInt;
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  case fieldCode of
    1 .. 2:
      begin
        while (checkErrorCode > 0) or (checkInt = 0) do
        begin
          ClearScreen();
          case fieldCode of
            1:
              write('Введите код комплектующего: ');
            2:
              write('Введите код типа комплектующего: ');
          end;
          readln(checkInput);
          writeln;
          val(string(checkInput), checkInt, checkErrorCode);
          if (checkErrorCode > 0) or (checkInt = 0) then
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
      end;
    3 .. 4:
      begin
        while checkInput = '' do
        begin
          ClearScreen();
          case fieldCode of
            3:
              write('Введите имя изготовителя: ');
            4:
              write('Введите имя модели: ');
          end;
          readln(checkInput);
          writeln;
          if (checkInput = '') then
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
        checkInput := LowerCase(Trim(checkInput));
      end;
  end;
  isTableShown := false;
  while list <> nil do
  begin
    case fieldCode of
```

```
          1:
            comparator := (list^.partListInfo.partCode = checkInt);
          2:
            comparator := (list^.partListInfo.partTypeCode = checkInt);
          3:
            comparator := (AnsiLowerCase(Trim(list^.partListInfo.manufacturer))
              = checkInput);
          4:
            comparator := (AnsiLowerCase(Trim(list^.partListInfo.modelName))
              = checkInput);
        end;
        if comparator then
        begin
          if not isTableShown then
          begin
            writeln('Искомые записи.');
            writeln;
            writeln('-----------------------------------------------------------------------------------------------------------------------
----------------');
            writeln('| Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |
Параметры    |   Цена   |  Количество  |');
            writeln('-----------------------------------------------------------------------------------------------------------------------
----------------');
            isTableShown := true;
          end;
          writeln('|', list^.partListInfo.partCode:19, ' |',
            list^.partListInfo.partTypeCode:24, ' |',
            list^.partListInfo.manufacturer:19, ' |',
            list^.partListInfo.modelName:19, ' |', list^.partListInfo.parameters
            :19, ' |', list^.partListInfo.price:9:2, ' |',
            list^.partListInfo.availability:13, ' |');
          writeln('-----------------------------------------------------------------------------------------------------------------------
----------------');
        end;
        list := list^.partListNextElement;
      end;
      if not isTableShown then
        writeln('Записи не найдены.');
      writeln;
      writeln('Нажмите, чтобы продолжить.');
      readln;
      isNotExitCheck := true;
      while isNotExitCheck do
      begin
        ClearScreen();
        write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения поиска: ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt, checkErrorCode);
        if checkErrorCode > 0 then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end
        else
          case checkInt of
            1:
              begin
                isNotExitCheck := false;
                list := header;
              end;
            0:
              begin
```

```
            isNotExitCheck := false;
            isNotExit := false;
          end;
        else
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
    end;
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от поиска.');
    sleep(1200);
    isNotExit := false;
  end;
 end;
end;

{ procedure FindInPartTypeList }
procedure FindInPartTypeList(list: PartTypeListType);

var
 checkInput: TString;
 checkInt, checkErrorCode: integer;

var
 fieldCode: integer;
 isTableShown, isNotExit, isNotExitCheck, comparator: boolean;
 header: PartTypeListType;

begin
 header := list;
 isNotExit := true;
 comparator := false;
 while isNotExit do
 begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 2)) do
  begin
    ClearScreen();
    writeln('Доступные поля для поиска: ');
    writeln;
    writeln('1. Код типа комплектующего.');
    writeln('2. Название.');
    writeln;
    write('Введите номер поля(введите 0 для выхода): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 2)) then
    begin
      writeln('Введенное поле не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if checkInt <> 0 then
  begin
    fieldCode := checkInt;
```

```
checkInput := '';
checkInt := 0;
checkErrorCode := 1;
case fieldCode of
  1:
    begin
      while (checkErrorCode > 0) or (checkInt = 0) do
      begin
        ClearScreen();
        write('Введите код типа комплектующего: ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt, checkErrorCode);
        if (checkErrorCode > 0) or (checkInt = 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
    end;
  2:
    begin
      while checkInput = '' do
      begin
        ClearScreen();
        write('Введите название: ');
        readln(checkInput);
        writeln;
        if checkInput = '' then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
      checkInput := AnsiLowerCase(Trim(checkInput));
    end;
end;
isTableShown := false;
while list <> nil do
begin
  case fieldCode of
    1:
      comparator := (list^.partTypeListInfo.partTypeCode = checkInt);
    2:
      comparator :=
        (AnsiLowerCase(Trim(list^.partTypeListInfo.partTypeName))
        = checkInput);
  end;
  if comparator then
  begin
    if not isTableShown then
    begin
      writeln('Искомые записи.');
      writeln;
      writeln('-----------------------------------------------');
      writeln('| Код типа комплектующего |      Название      |');
      writeln('-----------------------------------------------');
      isTableShown := true;
    end;
    writeln('|', list^.partTypeListInfo.partTypeCode:24, ' |',
      list^.partTypeListInfo.partTypeName:19, ' |');
    writeln('-----------------------------------------------');
  end;
```

```
      list := list^.partTypeListNextElement;
    end;
    if not isTableShown then
      writeln('Записи не найдены.');
    writeln;
    writeln('Нажмите, чтобы продолжить.');
    readln;
    isNotExitCheck := true;
    while isNotExitCheck do
    begin
      ClearScreen();
      write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения поиска: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if (checkErrorCode <> 0) then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
      end
      else
        case checkInt of
          1:
            begin
              isNotExitCheck := false;
              list := header;
            end;
          0:
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
        else
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
    end
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от поиска.');
    sleep(1200);
    isNotExit := false;
  end;
 end;
end;

{ procedure FindInCompatiblePartList }
procedure FindInCompatiblePartList(list: CompatiblePartListType);

var
 checkInput: TString;
 checkInt, checkErrorCode: integer;

var
 isTableShown, isNotExit, isNotExitCheck: boolean;
 header: CompatiblePartListType;

begin
 header := list;
```

```
isNotExit := true;
while isNotExit do
begin
  checkInput := '';
  checkInt := 0;
  checkErrorCode := 1;
  while (checkErrorCode > 0) do
  begin
    ClearScreen();
    write('Введите код комплектующего(введите 0 для выхода): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if checkInt <> 0 then
  begin
    isTableShown := false;
    while list <> nil do
    begin
      if (list^.compatiblePartListInfo.firstPartCode = checkInt) or
        (list^.compatiblePartListInfo.secondPartCode = checkInt) then
      begin
        if not isTableShown then
        begin
          writeln('Искомые записи.');
          writeln;
          writeln('------------------------------------------------------------');
          writeln('| Код первого комплектующего | Код второго комплектующего |');
          writeln('------------------------------------------------------------');
          isTableShown := true;
        end;
        writeln('|', list^.compatiblePartListInfo.firstPartCode:27, ' |',
          list^.compatiblePartListInfo.secondPartCode:27, ' |');
        writeln('------------------------------------------------------------');
      end;
      list := list^.compatiblePartListNextElement;
    end;
    if not isTableShown then
      writeln('Записи не найдены.');
    writeln;
    writeln('Нажмите, чтобы продолжить.');
    readln;
    isNotExitCheck := true;
    while isNotExitCheck do
    begin
      ClearScreen();
      write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения поиска: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if checkErrorCode <> 0 then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
      end
      else
        case checkInt of
          1:
```

```
            begin
              isNotExitCheck := false;
              list := header;
            end;
          0:
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
        else
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
    end;
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от поиска.');
    sleep(1200);
    isNotExit := false;
  end;
 end;
end;

{ AddToListFunctions }
{ function AddToListMenu }
function AddToListMenu(): integer;

var
 checkInput: TString;
 checkInt, checkErrorCode: integer;

begin
 checkInput := '';
 checkInt := 0;
 checkErrorCode := 1;
 while ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3))) do
 begin
  ClearScreen();
  writeln('Вы выбрали функцию добавления данных в спискок.');
  writeln;
  writeln('Доступные для добавления данных списки: ');
  writeln('1. Список комплектующих.');
  writeln('2. Список типов комплектующих.');
  writeln('3. Список совместимых комплектующих.');
  writeln;
  write('Введите номер списка,в котором хотите провести добавление(или введите 0, чтобы выйти из
функции добавления): ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3))) then
  begin
    writeln('Списка с заданным номером не существует. Нажмите для повторного ввода.');
    readln;
  end;
 end;
 result := checkInt;
end;
```

```
{ procedure AddToPartList }
procedure AddToPartList(list: PartListType; checkList: PartTypeListType);

var
  isNotExit, isNotExitCheck, isInList, isAgreed: boolean;
  findStr: TString;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;
  checkReal: real;

var
  checkHeader1, header: PartListType;
  checkHeader2: PartTypeListType;

begin
  header := list;
  checkInt := 0;
  isAgreed := false;
  while list^.partListNextElement <> nil do
    list := list^.partListNextElement;
  isNotExit := true;
  while isNotExit do
  begin
    ClearScreen();
    writeln('Нажмите, чтобы ознакомиться с доступными типами копмлектующих.');
    readln;
    ShowPartTypeList(checkList);
    isInList := true;
    while isInList do
    begin
      checkInput := '';
      checkInt := 0;
      checkErrorCode := 1;
      while (checkErrorCode > 0) do
      begin
        ClearScreen();
        write('Введите код типа комлектующего(или введите 0, чтобы выйти из данной функции): ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt, checkErrorCode);
        if (checkErrorCode > 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
      if checkInt <> 0 then
      begin
        checkHeader2 := checkList;
        while ((checkHeader2^.partTypeListInfo.partTypeCode <> checkInt) and
          (checkHeader2^.partTypeListNextElement <> nil)) do
          checkHeader2 := checkHeader2^.partTypeListNextElement;
        if checkHeader2^.partTypeListInfo.partTypeCode <> checkInt then
        begin
          writeln('Такого типа комплектующего не существует. Нажмите для повторного ввода.');
          readln;
        end
        else
        begin
          isInList := false;
          isAgreed := true;
```

```
    end;
  end
  else
  begin
    isInList := false;
    isAgreed := false;
  end;
end;
if isAgreed then
begin
  inc(header^.lastID);
  new(list^.partListNextElement);
  list := list^.partListNextElement;
  list^.partListInfo.partCode := header^.lastID;
  list^.partListNextElement := nil;
  list^.partListInfo.partTypeCode := checkInt;
  checkInput := '';
  while checkInput = '' do
  begin
    write('Введите имя изготовителя: ');
    readln(checkInput);
    writeln;
    if (checkInput = '') then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
      ClearScreen();
    end;
  end;
  list^.partListInfo.manufacturer := checkInput;
  isInList := true;
  while isInList do
  begin
    checkInput := '';
    while checkInput = '' do
    begin
      write('Введите имя модели: ');
      readln(checkInput);
      writeln;
      if (checkInput = '') then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
        ClearScreen();
      end;
    end;
    findStr := AnsiLowerCase(Trim(checkInput));
    checkHeader1 := header;
    while ((AnsiLowerCase(Trim(checkHeader1^.partListInfo.modelName)) <>
      findStr) and (checkHeader1^.partListNextElement <> nil)) do
      checkHeader1 := checkHeader1^.partListNextElement;
    if (AnsiLowerCase(Trim(checkHeader1^.partListInfo.modelName)) = findStr)
    then
    begin
      writeln('Данная модель уже есть в списке. Нажмите для повторного ввода.');
      readln;
      ClearScreen();
    end
    else
      isInList := false;
  end;
  list^.partListInfo.modelName := checkInput;
  checkInput := '';
```

```
while checkInput = '' do
begin
  write('Введите параметры модели: ');
  readln(checkInput);
  writeln;
  if (checkInput = '') then
  begin
    writeln('Некорректный ввод. Нажмите для повторного ввода.');
    readln;
    ClearScreen();
  end;
end;
list^.partListInfo.parameters := checkInput;
checkInput := '';
checkErrorCode := 1;
checkReal := -1;
while (checkErrorCode > 0) or (checkReal < 0) do
begin
  write('Введите цену: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkReal, checkErrorCode);
  if (checkErrorCode > 0) or (checkReal < 0) then
  begin
    writeln('Некорректный ввод. Нажмите для повторного ввода.');
    readln;
    ClearScreen();
  end;
end;
list^.partListInfo.price := checkReal;
checkInput := '';
checkErrorCode := 1;
checkInt := -1;
while (checkErrorCode > 0) or (checkInt < 0) do
begin
  write('Введите количество: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if (checkErrorCode > 0) or (checkInt < 0) then
  begin
    writeln('Некорректный ввод. Нажмите для повторного ввода.');
    readln;
    ClearScreen();
  end;
end;
list^.partListInfo.availability := checkInt;
ClearScreen();
writeln('Запись была добавлена в список.');
sleep(1200);
isNotExitCheck := true;
while isNotExitCheck do
begin
  ClearScreen();
  write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения добавления: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if (checkErrorCode <> 0) then
  begin
    writeln('Некорректный ввод. Нажмите для повторного ввода.');
    readln;
  end
```

```
      else
        case checkInt of
          1:
            isNotExitCheck := false;
          0:
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
        else
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
    end;
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от добавления записи.');
    sleep(1200);
    isNotExit := false;
  end;
 end;
end;

{ procedure AddToPartTypeList }
procedure AddToPartTypeList(list: PartTypeListType);

var
 isNotExit, isNotExitCheck, isInList, isAgreed: boolean;

var
 checkInput: TString;
 checkErrorCode: integer;
 checkInt: integer;

var
 findStr: TString;
 checkHeader, header: PartTypeListType;

begin
 header := list;
 isAgreed := false;
 while list^.partTypeListNextElement <> nil do
  list := list^.partTypeListNextElement;
 isNotExit := true;
 while isNotExit do
 begin
  isInList := true;
  while isInList do
  begin
    checkHeader := header;
    checkInput := '';
    while (checkInput = '') and (checkInput <> '0') do
    begin
      ClearScreen();
      write('Введите название типа комлектующего(или введите 0 для выхода из фукнции): ');
      readln(checkInput);
      writeln;
      if (checkInput = '') and (checkInput <> '0') then
      begin
```

```
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if checkInput = '0' then
  begin
    isInList := false;
    isAgreed := false;
  end
  else
  begin
    findStr := AnsiLowerCase(Trim(checkInput));
    while (AnsiLowerCase(Trim(checkHeader^.partTypeListInfo.partTypeName))
      <> findStr) and (checkHeader^.partTypeListNextElement <> nil) do
      checkHeader := checkHeader^.partTypeListNextElement;
    if (AnsiLowerCase(Trim(checkHeader^.partTypeListInfo.partTypeName))
      = findStr) then
    begin
      writeln('Данный элемент уже есть в списке. Нажмите для повторного ввода.');
      readln;
    end
    else
      isInList := false;
    isAgreed := true;
  end;
end;
if isAgreed then
begin
  inc(header^.lastID);
  new(list^.partTypeListNextElement);
  list := list^.partTypeListNextElement;
  list^.partTypeListNextElement := nil;
  list^.partTypeListInfo.partTypeCode := header^.lastID;
  list^.partTypeListInfo.partTypeName := checkInput;
  ClearScreen();
  writeln('Запись была добавлена в список.');
  sleep(1200);
  isNotExitCheck := true;
  while isNotExitCheck do
  begin
    ClearScreen();
    write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения добавления: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode <> 0) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end
    else
      case checkInt of
        1:
          isNotExitCheck := false;
        0:
          begin
            isNotExitCheck := false;
            isNotExit := false;
          end;
      else
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
```

```
          end;
        end;
      end;
    end
    else
    begin
      ClearScreen();
      writeln('Вы отказались от добавления записи.');
      sleep(1200);
      isNotExit := false;
    end;
  end;
end;

{ procedure AddToCompatiblePartList }
procedure AddToCompatiblePartList(list: CompatiblePartListType;
  checkList: PartListType);

var
  isNotExit, isNotExitCheck, isInList, isAgreed: boolean;

var
  checkErrorCode: integer;
  checkInput: TString;
  checkInt, checkInt1, checkInt2, temp: integer;

var
  header, tempHeader, checkHeader1: CompatiblePartListType;
  checkHeader2: PartListType;

begin
  header := list;
  checkInt1 := 0;
  checkInt2 := 0;
  isNotExit := true;
  isAgreed := false;
  while isNotExit do
  begin
    ClearScreen();
    writeln('Нажмите, чтобы ознакомиться со списком комплектующих.');
    readln;
    ShowPartList(checkList);
    isInList := true;
    while isInList do
    begin
      checkInput := '';
      checkErrorCode := 1;
      checkInt1 := 0;
      while (checkErrorCode > 0) or (checkInt1 = 0) do
      begin
        ClearScreen();
        write('Введите код первого комплектующего: ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt1, checkErrorCode);
        if (checkErrorCode > 0) or (checkInt1 = 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end
      end;
      checkInput := '';
      checkErrorCode := 1;
```

```
checkInt2 := 0;
while (checkErrorCode > 0) or (checkInt1 = 0) do
begin
  write('Введите код второго комплектующего: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt2, checkErrorCode);
  if (checkErrorCode > 0) or (checkInt2 = 0) then
  begin
    writeln('Некорректный ввод. Нажмите для повторного ввода.');
    readln;
    ClearScreen();
  end
end;
checkInput := '';
checkErrorCode := 1;
while checkErrorCode > 0 do
begin
  ClearScreen();
  write('Чтобы подтвердить добавление, введите 1, иначе 0: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if (checkErrorCode <> 0) then
  begin
    if ((checkInt <> 0) or (checkInt <> 1)) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end;
  end
  else
    case checkInt of
      1:
        isAgreed := true;
      0:
        isAgreed := false;
    end;
end;
if isAgreed then
begin
  if checkInt1 > checkInt2 then
  begin
    temp := checkInt1;
    checkInt1 := checkInt2;
    checkInt2 := temp;
  end;
  checkHeader1 := header;
  while ((checkHeader1^.compatiblePartListInfo.firstPartCode <> checkInt1)
    and (checkHeader1^.compatiblePartListInfo.secondPartCode <> checkInt2)
    ) and (checkHeader1^.compatiblePartListNextElement <> nil) do
    checkHeader1 := checkHeader1^.compatiblePartListNextElement;
  if ((checkHeader1^.compatiblePartListInfo.firstPartCode = checkInt1) and
    (checkHeader1^.compatiblePartListInfo.secondPartCode = checkInt2))
  then
  begin
    writeln('Данная запись уже есть в списке. Нажмите для повторного ввода.');
    readln;
  end
  else
  begin
    checkHeader2 := checkList;
    while (checkHeader2^.partListInfo.partCode <> checkInt1) and
```

```
          (checkHeader2^.partListNextElement <> nil) do
          checkHeader2 := checkHeader2^.partListNextElement;
        if checkHeader2^.partListInfo.partCode <> checkInt1 then
        begin
          writeln('Комплектующего с таким кодом не существует. Нажмите для повторного ввода.');
          readln;
        end
        else
        begin
          checkHeader2 := checkList;
          while (checkHeader2^.partListInfo.partCode <> checkInt2) and
            (checkHeader2^.partListNextElement <> nil) do
            checkHeader2 := checkHeader2^.partListNextElement;
          if checkHeader2^.partListInfo.partCode <> checkInt2 then
          begin
            writeln('Комплектующего с таким кодом не существует. Нажмите для повторного ввода.');
            readln;
          end
          else if checkInt1 = checkInt2 then
          begin
            writeln('Комплектующее не может быть совместимым само с собой. Нажмите для повторного
ввода.');
            readln;
          end
          else
            isInList := false;
        end;
      end;
    end
    else
      isInList := false;
  end;
  if isAgreed then
  begin
    tempHeader := list^.compatiblePartListNextElement;
    new(list^.compatiblePartListNextElement);
    list^.compatiblePartListNextElement^.compatiblePartListInfo.firstPartCode
      := checkInt1;
    list^.compatiblePartListNextElement^.compatiblePartListInfo.secondPartCode
      := checkInt2;
    list^.compatiblePartListNextElement^.compatiblePartListNextElement :=
      tempHeader;
    ClearScreen();
    writeln('Запись была добавлена в список.');
    sleep(1200);
    isNotExitCheck := true;
    while isNotExitCheck do
    begin
      ClearScreen();
      write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения добавления: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if (checkErrorCode <> 0) then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
      end
      else
        case checkInt of
          1:
            isNotExitCheck := false;
          0:
```

```
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
          else
            begin
              writeln('Некорректный ввод. Нажмите для повторного ввода.');
              readln;
            end;
          end;
      end;
    end
    else
    begin
      ClearScreen();
      writeln('Вы отказались от добавления записи.');
      sleep(1200);
      isNotExit := false;
    end;
  end;
end;

{ DeleteFromListFunctions }
{ function DeleteFromListMenu }
function DeleteFromListMenu(): integer;

var
  checkInput: TString;
  checkErrorCode: integer;
  checkInt: integer;

begin
  checkErrorCode := 1;
  checkInput := '';
  checkInt := 0;
  while ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3))) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию удаления данных из списков.');
    writeln;
    writeln('Доступные для удаления списки: ');
    writeln('1. Список комплектующих.');
    writeln('2. Список типов комплектующих.');
    writeln('3. Список совместимых комплектующих.');
    writeln;
    write('Введите номер списка, из которого хотите удалять(или введите 0, чтобы выйти из функции
удаления): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3))) then
    begin
      writeln('Списка с заданным номером не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  result := checkInt;
end;

{ procedure DeleteFromPartList }
procedure DeleteFromPartList(list: PartListType;
  deleteList1: CompatiblePartListType);
```

```
var
  isNotExit, isNotExitCheck, isInList, isAgreed, flag: boolean;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

var
  partCode: integer;

var
  checkHeader1, temp1: PartListType;
  checkHeader2, temp2: CompatiblePartListType;

begin
  isNotExit := true;
  isAgreed := false;
  partCode := 0;
  checkInt := 0;
  checkHeader1 := nil;
  while isNotExit do
  begin
    ClearScreen();
    writeln('По нажатию клавиши будет выведен список.');
    writeln;
    writeln('Во время просмотра выберите запись, чтобы в дальнейшем ввести ее код для удаления.');
    readln;
    ShowPartList(list);
    isInList := true;
    while isInList do
    begin
      checkInput := '';
      checkInt := 0;
      checkErrorCode := 1;
      while (checkErrorCode > 0) do
      begin
        ClearScreen();
        write('Введите код комплектующего(или 0 для выхода из функции): ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt, checkErrorCode);
        if (checkErrorCode > 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
      if checkInt = 0 then
      begin
        isInList := false;
      end
      else
      begin
        checkHeader1 := list;
        flag := true;
        while (checkHeader1^.partListNextElement <> nil) and flag do
        begin
          if (checkHeader1^.partListNextElement^.partListInfo.partCode <>
            checkInt) then
            checkHeader1 := checkHeader1^.partListNextElement
          else
            flag := false;
        end;
```

```
      if not flag then
      begin
        partCode := checkHeader1^.partListNextElement^.partListInfo.partCode;
        isInList := false;
      end
      else
      begin
        writeln('Комплектующего с введенным кодом не существует. Нажмите для повторного ввода.');
        readln;
      end;
    end;
  end;
  if checkInt = 0 then
  begin
    ClearScreen();
    writeln('Вы отказались от удаления записи.');
    sleep(1200);
    isNotExit := false;
  end
  else
  begin
    checkInput := '';
    checkErrorCode := 1;
    while checkErrorCode > 0 do
    begin
      ClearScreen();
      write('Удаление записи может повлечь удаление записей из других списков. Для подтверждения
удаления введите 1, иначе 0: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if (checkErrorCode <> 0) then
      begin
        if ((checkInt <> 0) or (checkInt <> 1)) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end
      else
        case checkInt of
          1:
            isAgreed := true;
          0:
            isAgreed := false;
        end;
    end;
    if isAgreed then
    begin
      temp1 := checkHeader1^.partListNextElement^.partListNextElement;
      dispose(checkHeader1^.partListNextElement);
      checkHeader1^.partListNextElement := temp1;
      checkHeader2 := deleteList1;
      while checkHeader2^.compatiblePartListNextElement <> nil do
      begin
        if (checkHeader2^.compatiblePartListNextElement.
          compatiblePartListInfo.firstPartCode = partCode) or
          (checkHeader2^.compatiblePartListNextElement.compatiblePartListInfo.
          secondPartCode = partCode) then
        begin
          temp2 := checkHeader2^.compatiblePartListNextElement^.
            compatiblePartListNextElement;
          dispose(checkHeader2^.compatiblePartListNextElement);
```

```
          checkHeader2^.compatiblePartListNextElement := temp2;
        end
        else
          checkHeader2 := checkHeader2^.compatiblePartListNextElement;
      end;
      ClearScreen();
      writeln('Запись была удалена.');
      sleep(1200);
    end
    else
    begin
      ClearScreen();
      writeln('Вы отказались от удаления записи.');
      sleep(1200);
    end;
    isNotExitCheck := true;
    while isNotExitCheck do
    begin
      ClearScreen();
      write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения удаления: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if (checkErrorCode <> 0) then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
      end
      else
        case checkInt of
          1:
            isNotExitCheck := false;
          0:
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
        else
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
    end;
  end;
 end;
end;

{ procedure DeleteFromPartTypeList }
procedure DeleteFromPartTypeList(deleteList1: PartListType;
 list: PartTypeListType; deleteList2: CompatiblePartListType);

var
 isNotExit, isNotExitCheck, isInList, isAgreed, flag: boolean;

var
 checkInput: TString;
 checkInt, checkErrorCode: integer;

var
 partTypeCode, partCode: integer;

var
```

```
  checkHeader1, temp1: PartListType;
  checkHeader2, temp2: PartTypeListType;
  checkHeader3, temp3: CompatiblePartListType;

begin
  isNotExit := true;
  isAgreed := false;
  checkInt := 0;
  partTypeCode := 0;
  checkHeader2 := nil;
  while isNotExit do
  begin
    ClearScreen();
    writeln('По нажатию клавиши будет выведен список.');
    writeln;
    writeln('Во время просмотра выберите запись, чтобы в дальнейшем ввести ее код для удаления.');
    readln;
    ShowPartTypeList(list);
    isInList := true;
    while isInList do
    begin
      checkInput := '';
      checkInt := 0;
      checkErrorCode := 1;
      while (checkErrorCode > 0) do
      begin
        ClearScreen();
        write('Введите код типа комплектующего(или 0, чтобы выйти из функции): ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt, checkErrorCode);
        if (checkErrorCode > 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
      if checkInt = 0 then
      begin
        isInList := false;
      end
      else
      begin
        checkHeader2 := list;
        flag := true;
        while (checkHeader2^.partTypeListNextElement <> nil) and (flag) do
        begin
          if (checkHeader2^.partTypeListNextElement^.partTypeListInfo.
            partTypeCode <> checkInt) then
            checkHeader2 := checkHeader2^.partTypeListNextElement
          else
            flag := false;
        end;
        if not flag then
        begin
          partTypeCode := checkHeader2^.partTypeListNextElement^.
            partTypeListInfo.partTypeCode;
          isInList := false;
        end
        else
        begin
          writeln('Такого типа комплектующего не существует. Нажмите для повторного ввода.');
          readln;
```

```
      end;
    end;
  end;
  if checkInt = 0 then
  begin
    ClearScreen();
    writeln('Вы отказались от добавления записи.');
    sleep(1200);
    isNotExit := false;
  end
  else
  begin
    checkInput := '';
    checkErrorCode := 1;
    while checkErrorCode > 0 do
    begin
      ClearScreen();
      write('Удаление записи может повлечь удаление записей из других списков. Для подтверждения
удаления введите 1, иначе 0: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if (checkErrorCode <> 0) then
      begin
        if ((checkInt <> 0) or (checkInt <> 1)) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end
      else
        case checkInt of
          1:
            isAgreed := true;
          0:
            isAgreed := false;
        end;
    end;
    if isAgreed then
    begin
      temp2 := checkHeader2^.partTypeListNextElement^.partTypeListNextElement;
      dispose(checkHeader2^.partTypeListNextElement);
      checkHeader2^.partTypeListNextElement := temp2;
      checkHeader1 := deleteList1;
      while checkHeader1^.partListNextElement <> nil do
      begin
        if checkHeader1^.partListNextElement^.partListInfo.partTypeCode = partTypeCode
        then
        begin
          partCode := checkHeader1^.partListNextElement^.
            partListInfo.partCode;
          temp1 := checkHeader1^.partListNextElement^.partListNextElement;
          dispose(checkHeader1^.partListNextElement);
          checkHeader1^.partListNextElement := temp1;
          checkHeader3 := deleteList2;
          while checkHeader3^.compatiblePartListNextElement <> nil do
          begin
            if (checkHeader3^.compatiblePartListNextElement.
              compatiblePartListInfo.firstPartCode = partCode) or
              (checkHeader3^.compatiblePartListNextElement.
              compatiblePartListInfo.secondPartCode = partCode) then
            begin
              temp3 := checkHeader3^.compatiblePartListNextElement^.
```

```
              compatiblePartListNextElement;
            dispose(checkHeader3^.compatiblePartListNextElement);
            checkHeader3^.compatiblePartListNextElement := temp3;
          end
          else
            checkHeader3 := checkHeader3^.compatiblePartListNextElement;
        end;
      end
      else
        checkHeader1 := checkHeader1^.partListNextElement;
    end;
    ClearScreen();
    writeln('Запись была удалена.');
    sleep(1200);
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от удаления записи.');
    sleep(1200);
  end;
  isNotExitCheck := true;
  while isNotExitCheck do
  begin
    ClearScreen();
    write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения удаления: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode <> 0) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end
    else
      case checkInt of
        1:
          isNotExitCheck := false;
        0:
          begin
            isNotExitCheck := false;
            isNotExit := false;
          end;
      else
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
  end;
 end;
end;

{ procedure DeleteFromCompatiblePartList }
procedure DeleteFromCompatiblePartList(list: CompatiblePartListType);

var
  isNotExit, isNotExitCheck, isInList, isAgreed, flag: boolean;

var
  checkInput: TString;
  checkInt, checkInt1, checkInt2, checkErrorCode, temp1: integer;
```

```
var
  checkHeader, temp: CompatiblePartListType;

begin
  isNotExit := true;
  isAgreed := false;
  checkInt1 := 0;
  checkHeader := nil;
  while isNotExit do
  begin
    ClearScreen();
    writeln('По нажатию клавиши будет выведен список.');
    writeln;
    writeln('Во время просмотра выберите запись, чтобы в дальнейшем ввести ее код для удаления.');
    readln;
    ShowCompatiblePartList(list);
    isInList := true;
    while isInList do
    begin
      checkInput := '';
      checkInt1 := 0;
      checkErrorCode := 1;
      while (checkErrorCode > 0) do
      begin
        ClearScreen();
        write('Введите код первого комплектующего(или 0 для выхода из функции): ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt1, checkErrorCode);
        if (checkErrorCode > 0) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
      if checkInt1 = 0 then
      begin
        isInList := false;
      end
      else
      begin
        checkInput := '';
        checkInt2 := 0;
        checkErrorCode := 1;
        while (checkErrorCode > 0) or (checkInt2 = 0) do
        begin
          write('Введите код второго комплектующего: ');
          readln(checkInput);
          writeln;
          val(string(checkInput), checkInt2, checkErrorCode);
          if (checkErrorCode > 0) or (checkInt2 = 0) then
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
            ClearScreen();
          end;
        end;
        checkHeader := list;
        flag := true;
        if checkInt1 > checkInt2 then
        begin
          temp1 := checkInt1;
```

```
      checkInt1 := checkInt2;
      checkInt2 := temp1;
    end;
    while (checkHeader^.compatiblePartListNextElement <> nil) and flag do
    begin
      if (checkHeader^.compatiblePartListNextElement^.
        compatiblePartListInfo.firstPartCode = checkInt1) and
        (checkHeader^.compatiblePartListNextElement^.compatiblePartListInfo.
        secondPartCode = checkInt2) then
        flag := false
      else
        checkHeader := checkHeader^.compatiblePartListNextElement;
    end;
    if flag then
    begin
      writeln('Данная запись отсутсвует в списке. Нажмите для повторного ввода.');
      readln;
    end
    else
      isInList := false;
  end;
end;
if checkInt1 = 0 then
begin
  ClearScreen();
  writeln('Вы отказались от удаления.');
  sleep(1200);
  isNotExit := false;
end
else
begin
  checkInput := '';
  checkErrorCode := 1;
  while checkErrorCode > 0 do
  begin
    ClearScreen();
    write('Для подтверждения удаления введите 1, иначе 0: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode <> 0) then
    begin
      if ((checkInt <> 0) or (checkInt <> 1)) then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
      end;
    end
    else
      case checkInt of
        1:
          isAgreed := true;
        0:
          isAgreed := false;
      end;
  end;
  if isAgreed then
  begin
    temp := checkHeader^.compatiblePartListNextElement^.
      compatiblePartListNextElement;
    dispose(checkHeader^.compatiblePartListNextElement);
    checkHeader^.compatiblePartListNextElement := temp;
    ClearScreen();
```

```
      writeln('Запись была удалена.');
      sleep(1200);
    end
    else
    begin
      ClearScreen();
      writeln('Вы отказались от удаления записи.');
      sleep(1200);
    end;
    isNotExitCheck := true;
    while isNotExitCheck do
    begin
      ClearScreen();
      write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения удаления: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt, checkErrorCode);
      if (checkErrorCode <> 0) then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
      end
      else
        case checkInt of
          1:
            isNotExitCheck := false;
          0:
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
        else
          begin
            writeln('Некорректный ввод. Нажмите для повторного ввода.');
            readln;
          end;
        end;
    end;
  end;
end;

{ EditInListFunctions }
{ function EditInListMenu }
function EditInListMenu(): integer;

var
  checkInput: TString;
  checkErrorCode: integer;
  checkInt: integer;

begin
  checkErrorCode := 1;
  checkInput := '';
  checkInt := 0;
  while ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3))) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию редактирования данных в списке.');
    writeln;
    writeln('Доступные для редактирования списки: ');
    writeln('1. Список комплектующих.');
    writeln('2. Список типов комплектующих.');
```

```
    writeln('3. Список совместимых комплектующих.');
    writeln;
    write('Введите номер списка, в котором хотите редактировать(или введите 0, чтобы выйти из функции
редактирования): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > 3))) then
    begin
      writeln('Списка с заданным номером не существует. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  result := checkInt;
end;

{ procedure EditInPartList }
procedure EditInPartList(list: PartListType; checkList: PartTypeListType);

var
  isNotExit, isNotExitCheck, isInList, isAgreed, flag1, flag2: boolean;
  findStr: TString;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;
  checkReal: real;

var
  checkHeader1, header: PartListType;
  checkHeader2: PartTypeListType;

begin
  isNotExit := true;
  header := nil;
  checkInt := 0;
  flag2 := false;
  isAgreed := false;
  checkReal := 0;
  while isNotExit do
  begin
    ClearScreen();
    writeln('Нажмите, чтобы вывести список комплектующих и выберите код для редактирования.');
    readln;
    ShowPartList(list);
    isInList := true;
    while isInList do
    begin
      checkErrorCode := 1;
      checkInput := '';
      checkInt := 0;
      while checkErrorCode > 0 do
      begin
        ClearScreen();
        write('Введите код комплектующего(или 0 для выхода): ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt, checkErrorCode);
        if checkErrorCode > 0 then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
```

```
      end;
      if checkInt = 0 then
      begin
        isInList := false;
        isAgreed := false;
      end
      else
      begin
        header := list;
        while (header^.partListInfo.partCode <> checkInt) and
          (header^.partListNextElement <> nil) do
          header := header^.partListNextElement;
        if (header^.partListInfo.partCode <> checkInt) then
        begin
          writeln('Такого комплектующего не существует. Нажммите для повторного ввода.');
          readln;
        end
        else
        begin
          isInList := false;
          isAgreed := true;
        end;
      end;
    end;
    if isAgreed then
    begin
      isInList := true;
      while isInList do
      begin
        flag1 := false;
        flag2 := false;
        while (not flag1) and (not flag2) do
        begin
          ClearScreen();
          write('Введите код типа комплектующего(нажмите для перехода к следующему полю или введите 0
для выхода): ');
          readln(checkInput);
          writeln;
          if checkInput = '' then
            flag2 := true
          else
          begin
            val(string(checkInput), checkInt, checkErrorCode);
            if checkErrorCode = 0 then
              flag1 := true
            else
            begin
              writeln('Некорректный ввод. Нажмите для повторного ввода.');
              readln;
            end;
          end;
        end;
        if flag2 then
        begin
          isAgreed := true;
          isInList := false;
        end;
        if (flag1) and (checkInt = 0) then
        begin
          isInList := false;
          isAgreed := false;
        end;
        if (flag1) and (checkInt <> 0) then
```

```
  begin
    checkHeader2 := checkList;
    while (checkHeader2^.partTypeListInfo.partTypeCode <> checkInt) and
      (checkHeader2^.partTypeListNextElement <> nil) do
      checkHeader2 := checkHeader2^.partTypeListNextElement;
    if (checkHeader2^.partTypeListInfo.partTypeCode <> checkInt) then
    begin
      writeln('Такого типа комплектующего не существует. Нажммите для повторного ввода.');
      readln;
    end
    else
    begin
      isInList := false;
      isAgreed := true;
    end;
  end;
end;
if isAgreed then
begin
  if not flag2 then
    header^.partListInfo.partTypeCode := checkInt;
  write('Введите имя изготовителя(нажмите для перехода к следующему полю): ');
  readln(checkInput);
  writeln;
  if checkInput <> '' then
    header^.partListInfo.manufacturer := checkInput;
  isInList := true;
  while isInList do
  begin
    write('Введите имя модели(нажмите для перехода к следующему полю): ');
    readln(checkInput);
    writeln;
    if checkInput <> '' then
    begin
      findStr := AnsiLowerCase(Trim(checkInput));
      checkHeader1 := list;
      while (string(LowerCase(checkHeader1^.partListInfo.modelName)) <>
        findStr) and (checkHeader1^.partListNextElement <> nil) do
        checkHeader1 := checkHeader1^.partListNextElement;
      if string(LowerCase(checkHeader1^.partListInfo.modelName)) = findStr
      then
      begin
        writeln('Данная модель уже есть в списке. Нажмите для повторного ввода.');
        readln;
        ClearScreen();
      end
      else
        isInList := false;
    end
    else
      isInList := false;
  end;
  if checkInput <> '' then
    header^.partListInfo.modelName := checkInput;
  write('Введите параметры(нажмите для перехода к следующему полю): ');
  readln(checkInput);
  writeln;
  if checkInput <> '' then
    header^.partListInfo.parameters := checkInput;
  flag1 := false;
  flag2 := false;
  while (not flag1) and (not flag2) do
  begin
```

```
  write('Введите цену(нажмите для перехода к следующему полю): ');
  readln(checkInput);
  writeln;
  if checkInput = '' then
    flag1 := true
  else
  begin
    val(string(checkInput), checkReal, checkErrorCode);
    if (checkErrorCode = 0) and (checkReal >= 0) then
      flag2 := true
    else
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
      ClearScreen();
    end;
  end;
end;
if (flag2) and (not flag1) then
  header^.partListInfo.price := checkReal;
flag1 := false;
flag2 := false;
while (not flag1) and (not flag2) do
begin
  write('Введите количество(нажмите для перехода к следующему полю): ');
  readln(checkInput);
  writeln;
  if checkInput = '' then
    flag1 := true
  else
  begin
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode = 0) and (checkInt >= 0) then
      flag2 := true
    else
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
      ClearScreen();
    end;
  end;
end;
if (flag2) and (not flag1) then
  header^.partListInfo.availability := checkInt;
ClearScreen();
writeln('Запись отредактирована.');
sleep(1200);
isNotExitCheck := true;
while isNotExitCheck do
begin
  ClearScreen();
  write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения редактирования: ');
  readln(checkInput);
  writeln;
  val(string(checkInput), checkInt, checkErrorCode);
  if (checkErrorCode <> 0) then
  begin
    writeln('Некорректный ввод. Нажмите для повторного ввода.');
    readln;
  end
  else
    case checkInt of
      1:
```

```
            isNotExitCheck := false;
          0:
            begin
              isNotExitCheck := false;
              isNotExit := false;
            end;
          else
            begin
              writeln('Некорректный ввод. Нажмите для повторного ввода.');
              readln;
            end;
          end;
      end;
    end
    else
    begin
      ClearScreen();
      writeln('Вы отказались от редактирования записи.');
      sleep(1200);
      isNotExit := false;
    end;
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от редактирования записи.');
    sleep(1200);
    isNotExit := false;
  end;
 end;
end;

{ procedure EditInPartTypeList }
procedure EditInPartTypeList(list: PartTypeListType);

var
 checkInput: TString;
 checkInt, checkErrorCode: integer;

var
 isNotExit, isNotExitCheck, isInList, isAgreed: boolean;
 header: PartTypeListType;

begin
 header := nil;
 isAgreed := false;
 isNotExit := true;
 while isNotExit do
 begin
  ClearScreen();
  writeln('Нажмите, чтобы вывести список типов комплектующих и выберите код для редактирования.');
  readln;
  ShowPartTypeList(list);
  isInList := true;
  while isInList do
  begin
    checkErrorCode := 1;
    checkInput := '';
    checkInt := 0;
    while checkErrorCode > 0 do
    begin
      ClearScreen();
      write('Введите код типа комплектующего(или 0 для выхода): ');
```

```
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if checkErrorCode > 0 then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if checkInt = 0 then
  begin
    isInList := false;
    isAgreed := false;
  end
  else
  begin
    header := list;
    while (header^.partTypeListInfo.partTypeCode <> checkInt) and
      (header^.partTypeListNextElement <> nil) do
      header := header^.partTypeListNextElement;
    if (header^.partTypeListInfo.partTypeCode <> checkInt) then
    begin
      writeln('Такого типа комплектующего не существует. Нажммите для повторного ввода.');
      readln;
    end
    else
    begin
      isInList := false;
      isAgreed := true;
    end;
  end;
end;
if isAgreed then
begin
  ClearScreen();
  write('Введите название(нажмите для перехода к следующему полю): ');
  readln(checkInput);
  if checkInput <> '' then
    header^.partTypeListInfo.partTypeName := checkInput;
  ClearScreen();
  writeln('Запись отредактирована.');
  sleep(1200);
  isNotExitCheck := true;
  while isNotExitCheck do
  begin
    ClearScreen();
    write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения редактирования: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode <> 0) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end
    else
      case checkInt of
        1:
          isNotExitCheck := false;
        0:
          begin
            isNotExitCheck := false;
            isNotExit := false;
```

```
            end;
          else
            begin
              writeln('Некорректный ввод. Нажмите для повторного ввода.');
              readln;
            end;
          end;
      end;
    end
    else
    begin
      ClearScreen();
      writeln('Вы отказались от редактирования записи.');
      sleep(1200);
      isNotExit := false;
    end;
  end;
end;

{ procedure EditInCompatiblePartList }
procedure EditInCompatiblePartList(list: CompatiblePartListType;
  checkList: PartListType);

var
  checkInput: TString;
  checkInt, checkInt1, checkInt2, temp, checkErrorCode: integer;

var
  isNotExit, isNotExitCheck, isInList, isAgreed, flag1, flag2, isInListMain,
    comparator: boolean;
  fieldCode, node: integer;

var
  header1, checkHeader1: CompatiblePartListType;
  checkHeader2: PartListType;

begin
  node := 0;
  isAgreed := false;
  checkInt2 := 0;
  checkInt1 := 0;
  header1 := nil;
  flag2 := false;
  flag1 := false;
  isNotExit := true;
  while isNotExit do
  begin
    ClearScreen();
    writeln('Нажмите, чтобы вывести список совместимых комплектующих.');
    readln;
    ShowCompatiblePartList(list);
    isInList := true;
    while isInList do
    begin
      checkErrorCode := 1;
      while checkErrorCode > 0 do
      begin
        ClearScreen();
        write('Введите код первого комплектующего(или 0 для выхода): ');
        readln(checkInput);
        writeln;
        val(string(checkInput), checkInt1, checkErrorCode);
        if checkErrorCode > 0 then
```

```
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  if checkInt1 = 0 then
  begin
    isAgreed := false;
    isInList := false;
  end
  else
  begin
    checkErrorCode := 1;
    while checkErrorCode > 0 do
    begin
      write('Введите код второго комплектующего: ');
      readln(checkInput);
      writeln;
      val(string(checkInput), checkInt2, checkErrorCode);
      if checkErrorCode > 0 then
      begin
        writeln('Некорректный ввод. Нажмите для повторного ввода.');
        readln;
        ClearScreen();
      end;
    end;
    if checkInt1 > checkInt2 then
    begin
      temp := checkInt1;
      checkInt1 := checkInt2;
      checkInt2 := temp;
    end;
    header1 := list;
    while ((header1^.compatiblePartListInfo.firstPartCode <> checkInt1) and
      (header1^.compatiblePartListInfo.secondPartCode <> checkInt2)) and
      (header1^.compatiblePartListNextElement <> nil) do
      header1 := header1^.compatiblePartListNextElement;
    if ((header1^.compatiblePartListInfo.firstPartCode <> checkInt1) and
      (header1^.compatiblePartListInfo.secondPartCode <> checkInt2)) then
    begin
      writeln('Данная запись отсутстует в списке. Нажмите для повторного ввода.');
      readln;
    end
    else
    begin
      isInList := false;
      isAgreed := true;
    end;
  end;
end;
if isAgreed then
begin
  checkErrorCode := 1;
  checkInt := 0;
  while (checkErrorCode > 0) and ((checkInt <> 1) or (checkInt <> 2)) do
  begin
    ClearScreen();
    write('Введите, код какого комплектующего хотите редактировать(1 или 2): ');
    readln(checkInput);
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) and ((checkInt <> 1) or (checkInt <> 2)) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
```

```
        readln;
      end;
    end;
    fieldCode := checkInt;
    isInListMain := true;
    while isInListMain do
    begin
      isInList := true;
      while isInList do
      begin
        flag1 := false;
        flag2 := false;
        while (not flag1) and (not flag2) do
        begin
          ClearScreen();
          write('Введите код комплектующего(нажмите для перехода к следующему полю или введите 0 для
выхода): ');
          readln(checkInput);
          writeln;
          if checkInput = '' then
            flag2 := true
          else
          begin
            val(string(checkInput), checkInt, checkErrorCode);
            if checkErrorCode = 0 then
              flag1 := true
            else
            begin
              writeln('Некорректный ввод. Нажмите для повторного ввода.');
              readln;
            end;
          end;
        end;
        if flag2 then
        begin
          isAgreed := true;
          isInList := false;
        end;
        if (flag1) and (checkInt = 0) then
        begin
          isInList := false;
          isAgreed := false;
        end;
        if (flag1) and (checkInt <> 0) then
        begin
          checkHeader2 := checkList;
          while (checkHeader2^.partListInfo.partCode <> checkInt) and
            (checkHeader2^.partListNextElement <> nil) do
            checkHeader2 := checkHeader2^.partListNextElement;
          if (checkHeader2^.partListInfo.partCode <> checkInt) then
          begin
            writeln('Такого комплектующего не существует. Нажммите для повторного ввода.');
            readln;
          end
          else
          begin
            isInList := false;
            isAgreed := true;
          end;
        end;
      end;
      if flag2 then
      begin
```

```
    isAgreed := true;
    isInListMain := false;
  end;
  if (flag1) and (checkInt = 0) then
  begin
    isAgreed := false;
    isInListMain := false;
  end;
  if (flag1) and (checkInt <> 0) then
  begin
    comparator := false;
    case fieldCode of
      1:
        begin
          node := header1^.compatiblePartListInfo.secondPartCode
            + checkInt;
          comparator :=
            (header1^.compatiblePartListInfo.secondPartCode = checkInt);
        end;
      2:
        begin
          node := header1^.compatiblePartListInfo.firstPartCode +
            checkInt;
          comparator :=
            (header1^.compatiblePartListInfo.firstPartCode = checkInt);
        end;
    end;
    if comparator then
    begin
      writeln('Комплектующее не может быть совместимо само с собой. Нажмите для повторного ввода.');
      readln;
    end
    else
    begin
      checkHeader1 := list;
      while (node <> (checkHeader1^.compatiblePartListInfo.firstPartCode +
        checkHeader1^.compatiblePartListInfo.secondPartCode)) and
        (checkHeader1^.compatiblePartListNextElement <> nil) do
        checkHeader1 := checkHeader1^.compatiblePartListNextElement;
      if (node = (checkHeader1^.compatiblePartListInfo.firstPartCode +
        checkHeader1^.compatiblePartListInfo.secondPartCode)) then
      begin
        writeln('Такая запись уже есть списке. Нажмите для повторного ввода.');
        readln;
      end
      else
      begin
        isAgreed := true;
        isInListMain := false;
      end;
    end;
  end;
end;
if isAgreed then
begin
  if not flag2 then
    case fieldCode of
      1:
        begin
          if checkInt > header1^.compatiblePartListInfo.secondPartCode
          then
          begin
            temp := checkInt;
```

```
            checkInt := header1^.compatiblePartListInfo.secondPartCode;
            header1^.compatiblePartListInfo.secondPartCode := temp;
          end;
          header1^.compatiblePartListInfo.firstPartCode := checkInt;
        end;
      2:
        begin
          if checkInt < header1^.compatiblePartListInfo.firstPartCode then
          begin
            temp := checkInt;
            checkInt := header1^.compatiblePartListInfo.firstPartCode;
            header1^.compatiblePartListInfo.firstPartCode := temp;
          end;
          header1^.compatiblePartListInfo.secondPartCode := checkInt;
        end;
    end;
  ClearScreen();
  writeln('Запись отредактирована.');
  sleep(1200);
  isNotExitCheck := true;
  while isNotExitCheck do
  begin
    ClearScreen();
    write('Введите 0, чтобы перейти к меню списков, или 1 для продолжения редактирования: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode <> 0) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
    end
    else
      case checkInt of
        1:
          isNotExitCheck := false;
        0:
          begin
            isNotExitCheck := false;
            isNotExit := false;
          end;
      else
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
        end;
      end;
  end;
 end
 else
 begin
   ClearScreen();
   writeln('Вы отказались от редактирования записи.');
   sleep(1200);
   isNotExit := false;
 end;
end
else
begin
  ClearScreen();
  writeln('Вы отказались от редактирования записи.');
  sleep(1200);
  isNotExit := false;
```

```
    end;
  end;
end;

{ SpecialFunctions }
{ function SpecialFunctionsMenu }
function SpecialFunctionsMenu(isSpecFunCompleted, isNeededToUpdate: boolean): integer;

var
  checkInput: TString;
  checkInt, checkErrorCode, right: integer;

begin
  checkErrorCode := 1;
  checkInput := '';
  checkInt := 0;
  case isSpecFunCompleted of
    false:
      right := 2;
    true:
      right := 3;
  end;
  while ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > right))) do
  begin
    ClearScreen();
    writeln('Вы выбрали пункт специальных функций.');
    if isNeededToUpdate then
      writeln('Данные о комбинациях не актуальны, необходимо обновление.');
    writeln;
    writeln('Доступные специальные функции: ');
    writeln;
    writeln('1. Подбор всех возможных вариантов комплектации компьютера в заданном ценовом диапазоне.');
    writeln('2. Оформление заказа понравившегося варианта.');
    if isSpecFunCompleted then
      writeln('3. Просмотр подобранных комбинаций.');
    writeln;
    write('Выберите функцию, введя ее номер(0 для выхода): ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if ((checkErrorCode > 0) or ((checkInt < 0) or (checkInt > right))) then
    begin
      writeln('Выбор функции произведен некорректно. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  result := checkInt;
end;

{ function GetCompatiblePartsArray }
function GetCompatiblePartsArray(list: CompatiblePartListType): TCompPartsArr;

  function Search(arr: TcompPartsMtx; code: integer): integer;

  var
    ans, i, c: integer;
    flag: boolean;

  begin
    ans := -1;
    i := 0;
    c := length(arr) - 1;
    flag := true;
```

```
    while (i <= c) and (flag) do
      if arr[i][0] = code then
        flag := false
      else
        inc(i);
    if not flag then
      ans := i;
    result := ans;
  end;

var
  cmpPtsMtx: TcompPartsMtx;
  id1, id2, i, j, min, temp: integer;
  res: TCompPartsArr;

begin
  list := list^.compatiblePartListNextElement;
  while list <> nil do
  begin
    id1 := Search(cmpPtsMtx, list^.compatiblePartListInfo.firstPartCode);
    id2 := Search(cmpPtsMtx, list^.compatiblePartListInfo.secondPartCode);
    if id1 = -1 then
    begin
      SetLength(cmpPtsMtx, length(cmpPtsMtx) + 1);
      SetLength(cmpPtsMtx[length(cmpPtsMtx) - 1],
        length(cmpPtsMtx[length(cmpPtsMtx) - 1]) + 2);
      cmpPtsMtx[length(cmpPtsMtx) - 1][0] :=
        list^.compatiblePartListInfo.firstPartCode;
      cmpPtsMtx[length(cmpPtsMtx) - 1][1] :=
        list^.compatiblePartListInfo.secondPartCode;
    end
    else
    begin
      SetLength(cmpPtsMtx[id1], length(cmpPtsMtx[id1]) + 1);
      cmpPtsMtx[id1][length(cmpPtsMtx[id1]) - 1] :=
        list^.compatiblePartListInfo.secondPartCode;
    end;
    if id2 = -1 then
    begin
      SetLength(cmpPtsMtx, length(cmpPtsMtx) + 1);
      SetLength(cmpPtsMtx[length(cmpPtsMtx) - 1],
        length(cmpPtsMtx[length(cmpPtsMtx) - 1]) + 2);
      cmpPtsMtx[length(cmpPtsMtx) - 1][0] :=
        list^.compatiblePartListInfo.secondPartCode;
      cmpPtsMtx[length(cmpPtsMtx) - 1][1] :=
        list^.compatiblePartListInfo.firstPartCode;
    end
    else
    begin
      SetLength(cmpPtsMtx[id2], length(cmpPtsMtx[id2]) + 1);
      cmpPtsMtx[id2][length(cmpPtsMtx[id2]) - 1] :=
        list^.compatiblePartListInfo.firstPartCode;
    end;
    list := list^.compatiblePartListNextElement;
  end;
  if length(cmpPtsMtx) <> 0 then
  begin
    min := 0;
    for i := 1 to length(cmpPtsMtx) - 1 do
      if (length(cmpPtsMtx[i]) < length(cmpPtsMtx[min])) then
        min := i;
    for i := 0 to length(cmpPtsMtx[min]) - 2 do
      for j := i + 1 to length(cmpPtsMtx) - 1 do
```

```
      if cmpPtsMtx[min][j] < cmpPtsMtx[min][i] then
      begin
        temp := cmpPtsMtx[min][j];
        cmpPtsMtx[min][j] := cmpPtsMtx[min][i];
        cmpPtsMtx[min][i] := temp;
      end;
    result := cmpPtsMtx[min];
  end
  else
    result := res;
end;

{ procedure GetAllCombsIndex }
procedure GetAllCombsIndex(var IndexArr: TcompPartsMtx; n, m: integer);

  function NextSet(var arr: TCompPartsArr): boolean;

  var
    k, i, j: integer;
    res: boolean;

  begin
    k := m;
    res := false;
    i := k - 1;
    while (i >= 0) and (not res) do
    begin
      if arr[i] < n - k + i + 1 then
      begin
        inc(arr[i]);
        for j := i + 1 to k - 1 do
          arr[j] := arr[j - 1] + 1;
        res := true;
      end;
      dec(i);
    end;
    result := res;
  end;

  procedure Insert(arr: TCompPartsArr);

  var
    i: integer;

  begin
    SetLength(IndexArr, length(IndexArr) + 1);
    for i := 0 to m - 1 do
    begin
      SetLength(IndexArr[length(IndexArr) - 1],
        length(IndexArr[length(IndexArr) - 1]) + 1);
      IndexArr[length(IndexArr) - 1][length(IndexArr[length(IndexArr) - 1]) - 1]
        := arr[i];
    end;
  end;

var
  arr: TCompPartsArr;
  i: integer;

begin
  SetLength(arr, n);
  for i := 0 to n - 1 do
    arr[i] := i + 1;
```

```
  Insert(arr);
  if n >= m then
    while NextSet(arr) do
      Insert(arr);
end;

{ function SortAllCombs }
function SortCombs(list: PartListType; var IndexMtx: TcompPartsMtx;
  var avaliable: TCompPartsArrAvaliable): TcompPartsArrPrice;

var
  i, j: integer;
  temp1: real;
  sum: TcompPartsArrPrice;
  temp2: TCompPartsArr;
  head: PartListType;

begin
  head := list;
  SetLength(sum, length(IndexMtx));
  SetLength(avaliable, length(IndexMtx));
  for i := 0 to length(IndexMtx) - 1 do
  begin
    avaliable[i] := true;
    for j := 0 to length(IndexMtx[i]) - 1 do
    begin
      list := head^.partListNextElement;
      while (list <> nil) and (list^.partListInfo.partCode <> IndexMtx[i][j]) do
        list := list^.partListNextElement;
      if list <> nil then
        sum[i] := sum[i] + list^.partListInfo.price;
    end;
  end;
  for i := 0 to length(sum) - 2 do
    for j := i + 1 to length(sum) - 1 do
      if sum[j] < sum[i] then
      begin
        temp1 := sum[j];
        sum[j] := sum[i];
        sum[i] := temp1;
        temp2 := IndexMtx[j];
        IndexMtx[j] := IndexMtx[i];
        IndexMtx[i] := temp2;
      end;
  for i := 0 to length(IndexMtx) - 1 do
    for j := 0 to length(IndexMtx[i]) - 1 do
    begin
      list := head^.partListNextElement;
      while (list <> nil) and (list^.partListInfo.partCode <> IndexMtx[i][j]) do
        list := list^.partListNextElement;
      if list <> nil then
        if list^.partListInfo.availability = 0 then
          avaliable[i] := false;
    end;

  result := sum;
end;

{ procedure ShowAllCombs }
function ShowAllCombs(list: PartListType; mtx: TcompPartsMtx;
  sum: TcompPartsArrPrice; price: real): integer;

var
```

```
  i, j, size: integer;
  head: PartListType;
  path: string;
  fl: TCombsFile;

begin
  path := ' ';
  size := 0;
  head := list;
  ClearScreen();
  repeat
    writeln('Введите путь, в котором хотите создать текстовый файл(Нажмите, чтобы не сохранять).');
    writeln;
    readln(path);
    if not directoryExists(path) and (path <> '') then
    begin
      writeln('Введенной директории не существует. Нажмите для повторного ввода.');
      readln;
      ClearScreen();
    end;
  until (path = '') or (directoryExists(path));
  if path <> '' then
  begin
    path := path + '\CombinationsFile_upozn.txt';
{$I-}
    assignFile(fl, path);
    rewrite(fl);
    writeln(fl, 'Все подобранные варианты.');
    writeln(fl);
    writeln(fl,
      '-------------------------------------------------------------------------------------------------------------------------------------------------------
------------------');
    writeln(fl,
      '| Номер | Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |
Параметры     |   Цена   |  Количество  | Сумма |');
    writeln(fl,
      '-------------------------------------------------------------------------------------------------------------------------------------------------------
------------------');
    for i := 0 to length(mtx) - 1 do
    begin
      for j := 0 to length(mtx[i]) - 1 do
      begin
        list := head^.partListNextElement;
        while (list <> nil) and (list^.partListInfo.partCode <> mtx[i][j]) do
          list := list^.partListNextElement;
        if (list <> nil) and (sum[i] <= price + 0.000001) then
        begin
          write(fl, '|', (i + 1):7, '|');
          write(fl, list^.partListInfo.partCode:19, ' |',
            list^.partListInfo.partTypeCode:24, ' |',
            list^.partListInfo.manufacturer:19, ' |',
            list^.partListInfo.modelName:19, ' |', list^.partListInfo.parameters
            :19, ' |', list^.partListInfo.price:9:2, ' |',
            list^.partListInfo.availability:13, ' |', sum[i]:6:2, ' |');
          writeln(fl);
        end;
      end;
      if (list <> nil) and (sum[i] <= price + 0.000001) then
        writeln(fl,
          '-------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------');
    end;
    closeFile(fl);
```

```
    ClearScreen();
    writeln('Информация сохранена в файл.');
    sleep(1200);
  end;
  ClearScreen();
  writeln('Все подобранные варианты.');
  writeln;
  writeln('-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------');
  writeln('| Номер | Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |
Параметры    |   Цена   | Количество  | Сумма |');
  writeln('-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------');
  for i := 0 to length(mtx) - 1 do
  begin
    for j := 0 to length(mtx[i]) - 1 do
    begin
      list := head^.partListNextElement;
      while (list <> nil) and (list^.partListInfo.partCode <> mtx[i][j]) do
        list := list^.partListNextElement;
      if (list <> nil) and (sum[i] <= price + 0.000001) then
      begin
        inc(size);
        write('|', (i + 1):7, '|');
        write(list^.partListInfo.partCode:19, ' |',
          list^.partListInfo.partTypeCode:24, ' |',
          list^.partListInfo.manufacturer:19, ' |', list^.partListInfo.modelName
          :19, ' |', list^.partListInfo.parameters:19, ' |',
          list^.partListInfo.price:9:2, ' |', list^.partListInfo.availability
          :13, ' |', sum[i]:6:2, ' |');
        writeln;
      end;
    end;
    if (list <> nil) and (sum[i] <= price + 0.000001) then
      writeln('-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------');
  end;
  writeln;
  writeln('Нажмите, чтобы продолжить.');
  readln;
  result := size;
end;

{ function MakeOrder }
function MakeOrder(list: PartListType; mtx: TcompPartsMtx;
  sum: TcompPartsArrPrice; avaliable: TCompPartsArrAvaliable; size: integer;
  price: real; isNeededToUpdate: boolean): boolean;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

var
  i, j, k, c: integer;
  head, toBuy, head1, temp: PartListType;
  fl: TCombsFile;
  path: string;

begin
  result := isNeededToUpdate;
  ClearScreen();
  head := list;
  k := 0;
```

```
  writeln('Вы выбрали функцию оформления заказа.');
  writeln;
  writeln('Все подобранные варианты, возможные для заказа.');
  writeln;
  writeln('-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------');
  writeln('| Номер | Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |
Параметры     |   Цена   | Количество  | Сумма |');
  writeln('-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------');
  for i := 0 to length(mtx) - 1 do
  begin
    for j := 0 to length(mtx[i]) - 1 do
    begin
      list := head^.partListNextElement;
      while (list <> nil) and (list^.partListInfo.partCode <> mtx[i][j]) do
        list := list^.partListNextElement;
      if (list <> nil) and (sum[i] <= price + 0.000001) and (avaliable[i]) then
      begin
        write('|', (k + 1):7, '|');
        write(list^.partListInfo.partCode:19, ' |',
          list^.partListInfo.partTypeCode:24, ' |',
          list^.partListInfo.manufacturer:19, ' |', list^.partListInfo.modelName
          :19, ' |', list^.partListInfo.parameters:19, ' |',
          list^.partListInfo.price:9:2, ' |', list^.partListInfo.availability
          :13, ' |', sum[i]:6:2, ' |');
        writeln;
      end;
    end;
    if (list <> nil) and (sum[i] <= price + 0.000001) and (avaliable[i]) then
    begin
      writeln('-------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------');
      inc(k);
    end;
  end;
  writeln;
  writeln('Нажмите, чтобы продолжить.');
  readln;
  ClearScreen();
  checkErrorCode := 1;
  checkInt := -1;
  while (checkErrorCode > 0) or ((checkInt < 0 - 1) or (checkInt >= k)) do
  begin
    write('Введите номер понравившегося варианта(или 0 для выхода): ');
    readln(checkInput);
    writeln;
    val(checkInput, checkInt, checkErrorCode);
    dec(checkInt);
    if (checkErrorCode > 0) or ((checkInt < -1) or (checkInt >= k)) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода.');
      readln;
      ClearScreen();
    end;
  end;
  if checkInt <> -1 then
  begin
    c := 0;
    new(toBuy);
    head1 := toBuy;
    toBuy^.partListNextElement := nil;
    for i := 0 to length(mtx) - 1 do
```

```
    begin
      for j := 0 to length(mtx[i]) - 1 do
      begin
        list := head^.partListNextElement;
        while (list <> nil) and (list^.partListInfo.partCode <> mtx[i][j]) do
          list := list^.partListNextElement;
        if (list <> nil) and (sum[i] <= price + 0.000001) and (avaliable[i])
        then
        begin
          if c = checkInt then
          begin
            new(toBuy^.partListNextElement);
            toBuy := toBuy^.partListNextElement;
            toBuy^.partListInfo := list^.partListInfo;
            dec(list^.partListInfo.availability);
            toBuy^.partListNextElement := nil;
            k := c;
          end;
        end;
      end;
      if (list <> nil) and (sum[i] <= price + 0.000001) and (avaliable[i]) then
      begin
        inc(c);
      end;
    end;
    ClearScreen();
    writeln('Ваш заказ.');
    writeln;
    writeln('----------------------------------------------------------------------------------------------------------------------------
----------');
    writeln('| Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |
Параметры    |  Цена  | Количество |');
    writeln('----------------------------------------------------------------------------------------------------------------------------
----------');
    toBuy := head1;
    head1 := head1^.partListNextElement;
    while head1 <> nil do
    begin
      write('|', head1^.partListInfo.partCode:19, ' |',
        head1^.partListInfo.partTypeCode:24, ' |',
        head1^.partListInfo.manufacturer:19, ' |', head1^.partListInfo.modelName
        :19, ' |', head1^.partListInfo.parameters:19, ' |',
        head1^.partListInfo.price:9:2, ' |', head1^.partListInfo.availability * 0 + 1
        :13, ' |');
      writeln;
      head1 := head1^.partListNextElement;
    end;
    writeln('----------------------------------------------------------------------------------------------------------------------------
----------');
    writeln;
    writeln('Сумма заказа: ', sum[k]:0:2);
    writeln;
    repeat
      writeln('Введите путь, в котором хотите создать текстовый файл.');
      writeln;
      readln(path);
      if not directoryExists(path) then
      begin
        writeln('Введенной директории не существует. Нажмите для повторного ввода.');
        readln;
        ClearScreen();
      end;
    until (directoryExists(path));
```

```
    path := path + '\OrderFile_upozn.txt';
  {$I-}
    assignFile(fl, path);
    rewrite(fl);
    head1 := toBuy;
    writeln(fl, 'Ваш заказ.');
    writeln(fl);
    writeln(fl,
      '-------------------------------------------------------------------------------------------------------------------------------------------
--');
    writeln(fl,
      '| Код комплектующего | Код типа комплектующего |    Изготовитель    |       Модель       |       Параметры
|   Цена   |  Количество  |');
    writeln(fl,
      '-------------------------------------------------------------------------------------------------------------------------------------------
--');
    head1 := head1^.partListNextElement;
    while head1 <> nil do
    begin
      write(fl, '|', head1^.partListInfo.partCode:19, ' |',
        head1^.partListInfo.partTypeCode:24, ' |',
        head1^.partListInfo.manufacturer:19, ' |', head1^.partListInfo.modelName
        :19, ' |', head1^.partListInfo.parameters:19, ' |',
        head1^.partListInfo.price:9:2, ' |', head1^.partListInfo.availability * 0 + 1
        :13, ' |');
      writeln(fl);
      head1 := head1^.partListNextElement;
    end;
    writeln(fl,
      '-------------------------------------------------------------------------------------------------------------------------------------------
--');
    writeln(fl);
    writeln(fl, 'Сумма заказа: ', sum[k]:0:2);
    closeFile(fl);
    ClearScreen();
    writeln('Информация записана в файл.');
    sleep(1200);
    result := true;
    while toBuy^.partListNextElement <> nil do
    begin
      temp := toBuy^.partListNextElement^.partListNextElement;
      dispose(toBuy^.partListNextElement);
      toBuy^.partListNextElement := temp;
    end;
    dispose(toBuy);
  end
  else
  begin
    ClearScreen();
    writeln('Вы отказались от покупки.');
    sleep(1200);
  end;
end;

{ ExitFunctions }
{ function SaveWithoutChanges }
function SaveWithoutChanges(list1: PartListType; list2: PartTypeListType;
  list3: CompatiblePartListType): boolean;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;
```

```
var
  executionContinue: boolean;
  temp1: PartListType;
  temp2: PartTypeListType;
  temp3: CompatiblePartListType;

begin
  checkErrorCode := 1;
  checkInt := -1;
  checkInput := '';
  while (checkErrorCode > 0) and ((checkInt <> 0) or (checkInt <> 1)) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию сохранения без изменений.');
    writeln;
    write('Введите 1, чтобы продолжить, или 0 для выхода из процедуры: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) and ((checkInt <> 0) or (checkInt <> 1)) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода');
      readln;
    end;
  end;
  if checkInt = 0 then
  begin
    ClearScreen();
    writeln('Вы отказались от выхода из программы.');
    sleep(1200);
    executionContinue := true;
  end
  else
  begin
    while list1^.partListNextElement <> nil do
    begin
      temp1 := list1^.partListNextElement^.partListNextElement;
      dispose(list1^.partListNextElement);
      list1^.partListNextElement := temp1;
    end;
    while list2^.partTypeListNextElement <> nil do
    begin
      temp2 := list2^.partTypeListNextElement^.partTypeListNextElement;
      dispose(list2^.partTypeListNextElement);
      list2^.partTypeListNextElement := temp2;
    end;
    while list3^.compatiblePartListNextElement <> nil do
    begin
      temp3 := list3^.compatiblePartListNextElement^.
        compatiblePartListNextElement;
      dispose(list3^.compatiblePartListNextElement);
      list3^.compatiblePartListNextElement := temp3;
    end;
    executionContinue := false;
  end;
  result := executionContinue;
end;

{ function SaveWithChanges }
function SaveWithChanges(list1: PartListType; list2: PartTypeListType;
  list3: CompatiblePartListType): boolean;

  function FileWriting(path1, path2, path3: string; list1: PartListType;
```

```
    list2: PartTypeListType; list3: CompatiblePartListType): boolean;

  var
    partListFile: PartListFileType;
    partTypeListFile: PartTypeListFileType;
    compatiblePartListFile: CompatiblePartListFileType;

  var
    temp1: PartListType;
    temp2: PartTypeListType;
    temp3: CompatiblePartListType;

  begin
{$I-}
    assign(partListFile, path1);
    rewrite(partListFile);
    while list1^.partListNextElement <> nil do
    begin
      write(partListFile, list1^.partListNextElement^.partListInfo);
      temp1 := list1^.partListNextElement^.partListNextElement;
      dispose(list1^.partListNextElement);
      list1^.partListNextElement := temp1;
    end;
    close(partListFile);
    assign(partTypeListFile, path2);
    rewrite(partTypeListFile);
    while list2^.partTypeListNextElement <> nil do
    begin
      write(partTypeListFile, list2^.partTypeListNextElement^.partTypeListInfo);
      temp2 := list2^.partTypeListNextElement^.partTypeListNextElement;
      dispose(list2^.partTypeListNextElement);
      list2^.partTypeListNextElement := temp2;
    end;
    close(partTypeListFile);
    assign(compatiblePartListFile, path3);
    rewrite(compatiblePartListFile);
    while list3^.compatiblePartListNextElement <> nil do
    begin
      write(compatiblePartListFile,
        list3^.compatiblePartListNextElement^.compatiblePartListInfo);
      temp3 := list3^.compatiblePartListNextElement^.
        compatiblePartListNextElement;
      dispose(list3^.compatiblePartListNextElement);
      list3^.compatiblePartListNextElement := temp3;
    end;
    close(compatiblePartListFile);
    result := false;
  end;

var
  checkInput: TString;
  checkInt, checkErrorCode: integer;

var
  directoryPath, folder_files_name, path1, path2, path3: string;
  error1, error2, error3: integer;
  executionContinue: boolean;

var
  partListFile: PartListFileType;
  partTypeListFile: PartTypeListFileType;
  compatiblePartListFile: CompatiblePartListFileType;
```

```
begin
  checkErrorCode := 1;
  checkInt := -1;
  checkInput := '';
  while (checkErrorCode > 0) and ((checkInt <> 0) or (checkInt <> 1)) do
  begin
    ClearScreen();
    writeln('Вы выбрали функцию сохранения с изменениями.');
    writeln;
    write('Введите 1, чтобы продолжить, или 0 для выхода из процедуры: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if (checkErrorCode > 0) and ((checkInt <> 0) or (checkInt <> 1)) then
    begin
      writeln('Некорректный ввод. Нажмите для повторного ввода');
      readln;
    end;
  end;
  if checkInt = 0 then
  begin
    ClearScreen();
    writeln('Вы отказались от выхода из программы.');
    sleep(1200);
    executionContinue := true;
  end
  else
  begin
    repeat
      ClearScreen();
      writeln('Введите путь, в котором хотите создать папку с файлами:');
      writeln;
      readln(directoryPath);
      writeln;
      if not directoryExists(directoryPath) then
      begin
        writeln('Указанная вами директория не существует. Нажмите для повторного ввода.');
        readln;
      end;
    until directoryExists(directoryPath);
    writeln('Введите имя папки. которую хотите создать: ');
    writeln;
    readln(folder_files_name);
    writeln;
    directoryPath := directoryPath + '\' + folder_files_name;
    path1 := directoryPath + '\' + folder_files_name + '_PartListData.upozn';
    path2 := directoryPath + '\' + folder_files_name +
      '_PartTypeListData.upozn';
    path3 := directoryPath + '\' + folder_files_name +
      '_CompatiblePartListData.upozn';
    if directoryExists(directoryPath) then
    begin
{$I-}
      assignFile(partListFile, path1);
      reset(partListFile);
      error1 := IOResult;
      assignFile(partTypeListFile, path2);
      reset(partTypeListFile);
      error2 := IOResult;
      assignFile(compatiblePartListFile, path3);
      reset(compatiblePartListFile);
      error3 := IOResult;
      closeFile(partListFile);
```

```
    closeFile(partTypeListFile);
    closeFile(compatiblePartListFile);
    if (error1 = 0) and (error2 = 0) and (error3 = 0) then
    begin
      checkInput := '';
      checkInt := -1;
      checkErrorCode := 1;
      while (checkErrorCode > 0) and ((checkInt <> 0) or (checkInt <> 1)) do
      begin
        writeln('Папка с введенным названием уже существует.');
        writeln;
        write('Введите 1 для перезаписи данных, иначе 0: ');
        readln(checkInput);
        val(string(checkInput), checkInt, checkErrorCode);
        if (checkErrorCode > 0) and ((checkInt <> 0) or (checkInt <> 1)) then
        begin
          writeln('Некорректный ввод. Нажмите для повторного ввода.');
          readln;
          ClearScreen();
        end;
      end;
      if checkInt = 0 then
      begin
        ClearScreen();
        writeln('Вы отказались от перезаписи.');
        sleep(1200);
        executionContinue := true;
      end
      else
      begin
        executionContinue := FileWriting(path1, path2, path3, list1,
          list2, list3);
        ClearScreen();
        writeln('Данные перезаписаны.');
        sleep(1200);
      end;
    end
    else
    begin
      executionContinue := FileWriting(path1, path2, path3, list1,
        list2, list3);
      ClearScreen();
      writeln('Данные записаны по директории.');
      sleep(1200);
    end;
  end
  else
  begin
    createDir(directoryPath);
    executionContinue := FileWriting(path1, path2, path3, list1,
      list2, list3);
    ClearScreen();
    writeln('Новая папка была создана. Данные записаны.');
    sleep(1200);
  end;
 end;
 result := executionContinue;
end;

{ function MainMenu }
function MainMenu(): integer;

var
```

```
  checkInput: TString;
  checkErrorCode: integer;
  checkInt: integer;

begin
  checkErrorCode := 1;
  checkInput := '';
  checkInt := 0;
  while ((checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 10))) do
  begin
    ClearScreen();
    writeln('Меню выбора функций:');
    writeln;
    writeln('1. Чтение данных из файла.');
    writeln('2. Просмотр списков.');
    writeln('3. Сортировка списков.');
    writeln('4. Поиск данных с использованием фильтра.');
    writeln('5. Добавление данных в списки.');
    writeln('6. Удаление данных из списков.');
    writeln('7. Редактирование данных списков.');
    writeln('8. Подбор всех вариантов сборки компьютера в заданном ценовом диапазоне и оформление
заказа.');
    writeln('9. Выход из программы без сохранения изменений.');
    writeln('10. Выход из программы с сохранением изменений.');
    writeln;
    write('Выберите функцию, введя ее номер: ');
    readln(checkInput);
    writeln;
    val(string(checkInput), checkInt, checkErrorCode);
    if ((checkErrorCode > 0) or ((checkInt < 1) or (checkInt > 10))) then
    begin
      writeln('Выбор функции произведен некорректно. Нажмите для повторного ввода.');
      readln;
    end;
  end;
  result := checkInt;
end;

{ functions codes }
var
  mainMenuCode, showListCode, sortListCode, findInListCode, addToListCode,
    deleteFromListCode, editInListCode, specFunCode: integer;
  mainMenuContinue, isReadFromFile, showListContinue, sortListContinue,
    findInListContinue, addToListContinue, deleteFromListContinue,
    editInListContinue, specFunContinue, isSpecFunCompleted, isNeededToUpdate: boolean;

  { lists declaration }
var
  partList: PartListType;
  partTypeList: PartTypeListType;
  compatiblePartList: CompatiblePartListType;

  { SpecFuncs arrs and lists }
var
  cmpPtsArr: TCompPartsArr;
  IndexMtx: TcompPartsMtx;
  n, m, i, j: integer;
  key, checkInput: TString;
  checkErrorCode, size: integer;
  sum: TcompPartsArrPrice;
  avaliable: TCompPartsArrAvaliable;
  price: real;
```

```
begin
  { lists memory allocation }
  new(partList);
  new(partTypeList);
  new(compatiblePartList);

  partList^.partListNextElement := nil;
  partTypeList^.partTypeListNextElement := nil;
  compatiblePartList^.compatiblePartListNextElement := nil;

  partTypeList^.lastID := 0;
  partList^.lastID := 0;
  price := -1;
  size := 0;

  { init message }
  writeln('Добро пожаловать в каталог комплектующих. Нажмите, чтобы открыть меню.');
  readln;

  { working til exit }
  isReadFromFile := false;
  isSpecFunCompleted := false;
  isNeededToUpdate := false;
  mainMenuContinue := true;
  while mainMenuContinue do
  begin
    mainMenuCode := MainMenu();
    case mainMenuCode of
      1: { ReadFromFileProcedure }
        ReadFromFiles(partList, partTypeList, compatiblePartList,
          isReadFromFile);
      2: { ShowListFunctions }
        begin
          showListContinue := true;
          while showListContinue do
          begin
            showListCode := ShowListMenu();
            case showListCode of
              0:
                showListContinue := false;
              1:
                ShowPartList(partList);
              2:
                ShowPartTypeList(partTypeList);
              3:
                ShowCompatiblePartList(compatiblePartList);
            end;
          end;
        end;
      3: { SortListFunctions }
        begin
          sortListContinue := true;
          while sortListContinue do
          begin
            sortListCode := SortListMenu();
            case sortListCode of
              0:
                sortListContinue := false;
              1:
                SortPartList(partList);
              2:
                SortPartTypeList(partTypeList);
              3:
```

```
        SortCompatiblePartList(compatiblePartList);
      end;
    end;
  end;
4: { FindInListFunctions }
  begin
    findInListContinue := true;
    while findInListContinue do
    begin
      findInListCode := FindInListMenu();
      case findInListCode of
        0:
          findInListContinue := false;
        1:
          FindInPartList(partList);
        2:
          FindInPartTypeList(partTypeList);
        3:
          FindInCompatiblePartList(compatiblePartList);
      end;
    end;
  end;
5: { AddToListFunctions }
  begin
    addToListContinue := true;
    while addToListContinue do
    begin
      addToListCode := AddToListMenu();
      case addToListCode of
        0:
          addToListContinue := false;
        1:
          AddToPartList(partList, partTypeList);
        2:
          AddToPartTypeList(partTypeList);
        3:
          AddToCompatiblePartList(compatiblePartList, partList);
      end;
    end;
  end;
6: { DeleteFromListFunctions }
  begin
    deleteFromListContinue := true;
    while deleteFromListContinue do
    begin
      deleteFromListCode := DeleteFromListMenu();
      case deleteFromListCode of
        0:
          deleteFromListContinue := false;
        1:
          DeleteFromPartList(partList, compatiblePartList);
        2:
          DeleteFromPartTypeList(partList, partTypeList,
            compatiblePartList);
        3:
          DeleteFromCompatiblePartList(compatiblePartList);
      end;
    end;
  end;
7: { EditInListFunctions }
  begin
    editInListContinue := true;
    while editInListContinue do
```

```
    begin
      editInListCode := EditInListMenu();
      case editInListCode of
        0:
          editInListContinue := false;
        1:
          EditInPartList(partList, partTypeList);
        2:
          EditInPartTypeList(partTypeList);
        3:
          EditInCompatiblePartList(compatiblePartList, partList);
      end;
    end;
  end;
8: { SpecialFunctions }
  begin
    specFunContinue := true;
    while specFunContinue do
    begin
      specFunCode := SpecialFunctionsMenu(isSpecFunCompleted, isNeededToUpdate);
      case specFunCode of
        0:
          specFunContinue := false;
        1:
          begin
            key := '';
            if isSpecFunCompleted then
            begin
              while ((key[1] <> '1') or (key[1] <> '0')) and
                (length(key) <> 1) do
              begin
                ClearScreen();
                writeln('Комбинации уже составлены. Желаете ли совершить перезапись?');
                write('Введите 1, если да, или 0, если нет: ');
                readln(key);
                if ((key[1] <> '1') or (key[1] <> '0')) and
                  (length(key) <> 1) then
                begin
                  writeln('Некорректный ввод. Нажмите для повторного ввода.');
                  readln;
                end;
              end;
            end;
            if (not isSpecFunCompleted) or (key = '1') then
            begin
              checkErrorCode := 1;
              price := -1;
              while (checkErrorCode > 0) and (price < 0) do
              begin
                ClearScreen();
                write('Введите ценовой диапазон: ');
                readln(checkInput);
                val(checkInput, price, checkErrorCode);
                if (checkErrorCode > 0) and (price < 0) then
                begin
                  writeln('Некорректный ввод. Нажмите для повторного ввода.');
                  readln;
                end;
              end;
              SetLength(cmpPtsArr, 0);
              cmpPtsArr := GetCompatiblePartsArray(compatiblePartList);
              if length(cmpPtsArr) = 0 then
              begin
```

```
                ClearScreen();
                writeln('Комбинаций не обнаружилось.');
                sleep(1200);
                isSpecFunCompleted := false;
              end
              else
              begin
                n := length(cmpPtsArr);
                SetLength(IndexMtx, 0);
                for m := 2 to n do
                  GetAllCombsIndex(IndexMtx, n, m);
                ClearScreen();
                writeln('Комбинации подобраны.');
                sleep(1200);
                for i := 0 to length(IndexMtx) - 1 do
                  for j := 0 to length(IndexMtx[i]) - 1 do
                    IndexMtx[i][j] := cmpPtsArr[IndexMtx[i][j] - 1];
                SetLength(sum, 0);
                setLength(avaliable, 0);
                sum := SortCombs(partList, IndexMtx, avaliable);
                isSpecFunCompleted := true;
                isNeededToUpdate := false;
              end;
            end;
          end;
        2:
          begin
            isNeededToUpdate := MakeOrder(partList, IndexMtx, sum, avaliable, size, price, isNeededToUpdate);
          end;
        3:
          begin
            if isSpecFunCompleted then
            begin
              ClearScreen();
              size := ShowAllCombs(partList, IndexMtx, sum, price);
            end;
          end;
      end;
    end;
  end;
  9: { SaveWithoutChanges function }
    mainMenuContinue := SaveWithoutChanges(partList, partTypeList,
      compatiblePartList);
  10: { SaveWithChanges function }
    mainMenuContinue := SaveWithChanges(partList, partTypeList,
      compatiblePartList);
 end;
end;
ClearScreen();
{ lists memory disposion }
dispose(partList);
dispose(partTypeList);
dispose(compatiblePartList);

{ bye message }
writeln('Спасибо, что воспользовались нашим приложением.');
sleep(930);

end.
```