Министерство науки и высшего образования Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

С. А. Фёдоров *А. В. Петров*

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ
ПРАКТИКУМ

Учебное пособие

Санкт-Петербург
2024

УДК 004.42

Фёдоров С. А. **Алгоритмы и структуры данных. Практикум** : учеб. пособие / С. А. Фёдоров, А. В. Петров. – СПб.: СПбПУ, 2023. – 32 с.

Учебное пособие соответствует основным разделам дисциплины «Алгоритмы и структуры данных».

Студентам предлагаются для выполнения лабораторные работы по различным структурам данных: массивам строк, массивам символов, структур массивов, массивам структур, однонаправленным и двунаправленным динамическим спискам и деревьям.

Учебное пособие предназначено для студентов, обучающихся по направлению подготовки бакалавров 09.03.04 «Программная инженерия», в частности, по онлайн-программе 09.03.04_03 «Разработка программного обеспечения».

Содержание

Введение

В данном пособии студентам, обучающимся на разработчиков программного обеспечения, предлагаются для выполнения лабораторные работы по различным структурам данных:

- массивам строк;
- массивам символов;
- структур массивов;
- массивам структур;
- однонаправленным динамическим спискам;
- двунаправленным динамическим спискам;
- деревьям.

При выполнении лабораторных работ, создавая своё *произведение*, необходимо следовать дисциплине программирования. Каждая работа оформляется как отдельный программный проект. При сдаче принимается исходный код ПО и результаты его работы.

§ 1. Указания к лабораторным работам

При выполнении лабораторных работ необходимо следовать следующим указаниям:

1. Сортировка динамических структур данных проводится *по ссылкам* (не по значениям элементов).
2. Ввод/вывод и обработка динамических структур данных проводится средствами *функционального* программирования – *рекурсивными хвостовыми процедурами*, все из которых являются *чистыми* (кроме процедур внешнего ввода/вывода).
3. При выводе динамических структур данных должна *гарантироваться* их целостность (соответствующие формальные аргументы должны иметь намерение *in*).
4. Использовать, где это необходимо, средства ООП (кроме л/р № 1): инкапсуляцию, наследование, полиморфизм, функции завершения.
5. Все рекурсивные процедуры должны быть *хвостовыми*. Но, например, чтобы реализовать уничтожение списка лучше использовать функцию завершения. Также можно применять хвостовую рекурсию, работая со списком как со стеком.

§ 2. Советы к лабораторным работам

Если ваш компилятор не поддерживает *в исходном коде* неименованные строковые константы в кодировке UTF-8 (например, gfortran: 45179), то при необходимости сравнения символа строки с определённым символом используется код этого символа (коды **в десятичном формате:** unicode-table.com):

```
if (string(1:1) == Char(1052, CH_) ! Сравнение 1-ого
символа с «М».
YES = Char(1044, CH_) // Char(1072, CH_) ! Запись в
константу YES слова «Да».
```

§ 3. Лабораторная работа № 1. Сплошные и несплошные структуры данных

3.1 Опорный вариант лабораторной работы № 1

Дан список группы с результатам сессии и с не вычисленным средним баллом в виде:

```
ФАМИЛИЯ  ИНИЦИАЛЫ ПОЛ     РЕЗУЛЬТАТЫ_СЕССИИ СРЕДНИЙ_БАЛЛ
15 симв. 5 симв.  1 симв. 5 симв.           3 симв.
```

Пример входного файла:

```
Дудиков         Д. Р. М 43453 0.0
Тихонов         Л. П. М 55353 0.0
Садовникова     П. О. Ж 43543 0.0
Степин          К. Д. М 55445 0.0
Воробъёва       Е. Р. Ж 44353 0.0
```

Рассчитать средний балл для каждого из учащихся. Отсортировать по убыванию среднего балла списки юношей и девушек по отдельности.

Пример выходного файла:

```
Успеваемость юношей:
Степин          К. Д. М 55445 4.60
Тихонов         Л. П. М 55353 4.20
Дудиков         Д. Р. М 43453 3.80
Успеваемость девушек:
Воробъёва       Е. Р. Ж 44353 3.80
Садовникова     П. О. Ж 43543 3.80
```

3.2 Допуски к проектам лабораторной работы

1. Собрать и запустить опорные лабораторные lab_1_1 –

lab_1_5.

2. Разобраться в их исходном коде.
3. Ответить на вопросы преподавателя по исходному коду.
4. Типичные вопросы:
    1. Для lab_1_1. Для чего MARKS_AMOUNT в формате при чтении указывается?
    2. Для lab_1_2. Для чего SURNAME_LEN в формате при чтении указывается?
    3. Для lab_1_2. Обработка данных в процедуре Get_list_by_gender проводится регулярным образом или нет?
    4. Для lab_1_3. Каково назначение recl (что это, в чём измеряется, чему равно и как именно влияет её значение на программу)?
    5. Для lab_1_4. Чем нужно руководствоваться при выборе «массива структур» или «структуры массивов»?
    6. Для lab_1_5. В обработке данных на строке 20 почему мы поставили условие именно N>=3?
    7. Для lab_1_6. Зачем на 31-ой строке в модуле обработки явное зануление ссылки? Когда оно не напрасно?

§ 4. Лабораторная работа № 2. Однонаправленные списки

4.1 Общая часть задания

Задание выполняется в виде программного проекта из двух модулей, в котором необходимо использовать *динамические однонаправленные списки*. Списки необходимо обрабатывать чистой хвостовой рекурсией **при всех** операциях с ними. При возможности применяется регулярное программирование.

4.2 Опорный вариант лабораторной работы № 2

Во входном файле F1 находится исходный код программы на Fortran, а в файле F2 — тот же код, но с добавлением некоторых строк. Сформировать файл из новых строк, пометив их в начале как «++ ».

Пример входного файла F1:

```
module environment
   use ISO_Fortran_Env

   implicit none

   integer, parameter      :: I_ = int16
   integer, parameter      :: C_ = R_
   character(*), parameter :: E_ = "UTF-8"

   interface operator (//)
      module procedure IntPlusString
   end interface
```

```
contains

   pure function IntPlusString(int, str) result(res)
      integer, intent(in)
:: int
      character(*), intent(in)
:: str
      character(len(str)+Floor(Log10(Real(int, real64)))
+1) :: res

      write (res,'(i0, a)') int, str
   end function IntPlusString

end module environment
```

Пример входного файла F2:

```
module environment
   use ISO_Fortran_Env

   implicit none

   integer, parameter      :: I_ = int16
   integer, parameter      :: R_ = real32
   integer, parameter      :: C_ = R_
   integer, parameter      :: CH_ =
Selected_Char_Kind("ISO_10646")
   character(*), parameter :: E_ = "UTF-8"

   interface operator (//)
      module procedure IntPlusString
      module procedure StringPlusInt
   end interface

contains
```

```
   pure function IntPlusString(int, str) result(res)
      integer, intent(in)
:: int
      character(*), intent(in)
:: str
      character(len(str)+Floor(Log10(Real(int, real64)))
+1) :: res

      write (res,'(i0, a)') int, str
   end function IntPlusString

   pure function StringPlusInt(str, int) result(res)
      character(*), intent(in)
:: str
      integer, intent(in)
:: int
      character(len(str)+Floor(Log10(Real(int, real64)))
+1)  :: res

      write (res,'(a, i0)') str, int
   end function StringPlusInt

end module environment

! Нечто.
```

Пример выходного файла:

```
++    integer, parameter      :: R_ = real32
++    integer, parameter      :: CH_ =
Selected_Char_Kind("ISO_10646")
++       module procedure StringPlusInt
++    pure function StringPlusInt(str, int) result(res)
++       character(*), intent(in)
:: str
++       integer, intent(in)
```

```
:: int
++       character(len(str)+Floor(Log10(Real(int,
real64)))+1)  :: res
++
++       write (res,'(a, i0)') str, int
++    end function StringPlusInt
++
++
++ ! Нечто.
```

4.3 Допуск к лабораторной работе

1. Собрать и запустить опорную лабораторную lab_2.
2. Разобраться в исходном коде.
3. Ответить на вопросы преподавателя по исходному коду.
4. Типичные вопросы:
   - Почему на 23 строке Initial_code идёт с next, а на 33 строке – без него?

4.4 Индивидуальные задания

1. Разработать чистую подпрограмму сдвига части строк в данном тексте влево или вправо на N символов, начиная со строки N1 до строки N2. Значения N, N1, N2, а также направление сдвига («left», «right») задаются во втором входном файле.

   Пример первого входного файла:

```
1
12
123
1234
12345
```

Пример второго входного файла:

```
2 3 4 L
```

Пример выходного файла:

```
1
12
3
34
12345
```

Указание. Элементом списка является строка. Для хранения строки лучше использовать размещаемый массив символов (EOShift).

2. Разработать чистую функцию проверки того, что данная строка A(M) состоит только из символов заданной строки B(L).

Указание. Элементом списка является символ строки. Возвращать номер символа, не найденного в строке B(L). Иначе возвращаем M+1.

3. Разработать чистую функцию формирования новой строки C(L+M) с помощью вставки заданной строки B(L) в строку A(M) после k-го символа ($L \le 10$, $M \le 10$, $0 \le k \le M$).

Указание. Элементом списка является символ строки.

4. Разработать чистую подпрограмму контекстного поиска и замены заданного слова S1 на слово S2 всюду в тексте.

Указание. Элементом списка может являться слово исходного файла или отдельная строка текста (Index, Len_Trim).

При хранении по словам: символ конца строки (перевод каретки) хранить в виде отдельного элемента списка, со-

держащего всего один символ, возвращаемый функцией New_line(CH_); слова читать во временную строку str, используя оператор

```
read (In, '(a)', advance='no', size=size, iostat=io) str
```

где size — число реально прочитанных символов. Вы дойдёте до конца строки, когда io == IOSTAT_EOR (константа, обозначающая чтение конца записи).

5. Разработать чистую подпрограмму перемещения в тексте группы строк со строки First по строку Last после K-ой строки. Значения First, Last, K задаются во втором входном файле. Правильными данными считается ситуация , когда First≤Last ,а K не принадлежит интервалу [First, Last].

   Указание. Элементом списка является строка.

6. Разработать чистую подпрограмму Top установки окна на начало текста и чистую подпрограмму Bottom установки на конец данного текста. Размер окна N (число строк в окне) задаётся во втором входном файле.

   Указание. Элементом списка является строка.

7. Разработать чистую подпрограмму удаления из текста группы строк с номерами от начальной First до конечной Last.

   Указание. Элементом списка является строка. Удаление проводить в созданном динамическом списке.

8. Разработать чистую подпрограмму вставки в строке на N-ое место M символов и чистую подпрограмму удаления в строке с N-го места M символов. После каждой вставки или удаления выводить обновленную строку. Ре-

жим ввода или удаления, а также число символов задаются во входном файле.

Пример входного файла:

```
Just for what?
D 10 5
I 10 fun!
```

Пример выходного файла:

```
Just for what?
D 10 5
Just for_
I 10 fun!
Just for fun!
```

Указание. Элементом списка является символ строки. Головная программа должна состоять из вызова процедур: чтение строки, вывод строки, а затем в цикле: чтение команды, выполнение команды, вывод строки.

9. Разработать чистую подпрограмму копирования группы строк данного текста от строки First до строки Last и вставки их после M-ой строки. Правильному набору вводимых данных соответствует ситуация, когда 0≤First≤Last и M не входит в диапазон [First, Last].

Указание. Элементом списка является строка.

10. Разработать чистую подпрограмму контекстной замены каждого отдельного вхождения подстроки B(L) в строку A(M) на L символов '*'. Если подстроки B(L) нет внутри A(M), либо ее принципиально нельзя выделить из A(M), то оставлять строку A без изменений.

Указание. Элементом списка является символ строки.

11. Разработать чистую подпрограмму пролистывания за-

данного текста. Размер листа N и направления пролистывания задаются во втором входном файле.

Пример первого входного файла:

```
1
2
3
4
5
```

Пример второго входного файла:

```
2
F
F
B
```

Пример выходного файла:

```
Исходный файл:
1
2
3
4
5

Размер листа:
2

Пролистывание:
1
2

F
2
3
```

F
3
4

B
2
3

Указание. Элементом первого списка является строка. Элементом второго списка является символ F или B. Длину окна N храним в отдельной переменной.

12. Разработать чистую функцию поиска подстроки B(L) в строке A(M).

Указание. Элементом списка является символ строки. Функция возвращает номер позиции и 0 в случае ненахождения вхождения.

13. Разработать чистую функцию выделения подстроки B(L) из строки A(M), начиная с её K-ого символа. Выделять столько символов, сколько возможно, вплоть до ни одного.

Указание. Элементом списка является символ строки.

14. Разработать чистую процедуру сортировки строк заданного текста по убыванию длины строки. Использовать сортировку вставками.

Указание. Элементом списка является строка. Применять функцию Len_trim.

15. Разработать чистую подпрограмму “центрирования” строк заданного текста, каждая из которых имеет длину $L \le 40$ символов и строки имеют разное число пробелов слева.

Указание. Элементом списка является размещаемая

строка.

16. Разработать чистую подпрограмму сортировки списка имен файлов с расширениями по имени и чистую подпрограмму сортировки того же списка по расширению. Использовать метод сортировки выбором.

Указание. Элемент списка содержит название файла и его расширение. Применять функцию Index для поиска точки после предварительного прочтения всего имени файла.

17. Разработать чистую подпрограмму удаления из списка файлов имён с заданным расширением.

Пример входного файла:

```
obj
main.f90
main.obj
group.f90
group.obj
```

Пример выходного файла:

```
Исходный список:
obj
main.f90
main.obj
group.f90
group.obj

Полученный список:
main.f90
group.f90
```

Указание. Элемент списка содержит название файла и его расширение. Применять функцию Index для поиска

точки после предварительного прочтения всего имени файла.

18. Разработать чистую подпрограмму поиска в списке файлов имён по заданной маске. Например, если вводится строка wi*.* , то необходимо оставить имена файлов с любыми расширениями, но содержащие в качестве первых букв имени буквы wi.

Указания.

- Элемент списка содержит название файла и его расширение. Применять функцию Index для поиска точки после предварительного прочтения всего имени файла.
- В разрабатываемой подпрограмме лучше сперва получить список, удовлетворяющий части маски по имени, а потом из него — список, удовлетворяющий части маски по расширению.
- Проверку соответствия части маски (по имени или по расширению) можно проводить, проверяя сперва соответствие начала, а потом соответствие конца этой части. Для этого каждую часть маски можно хранить как две строки. Например, маска wi*.* представляется такими двумя частями: («wi», «») и («», «»). Тогда при проверке части маски по имени нужно найти все файлы, первые два символа которых совпадают с «wi», а последние — не важны (длина «» равна 0), а при проверке части маски по расширению нужно найти все файлы, первые и последние символы которых не важны (длина «» и «» равна 0).

19. Дан список имен файлов с расширениями, расположен-

ных по алфавиту, и список ранее удаленных файлов в произвольном порядке. Разработать чистую подпрограмму восстановления в первом списке имен файлов из второго списка, вставляя имена из последнего в нужное место по алфавиту.

Указание. Элемент списка содержит название файла и его расширение. Применять функцию Index для поиска точки после предварительного прочтения всего имени файла.

20. Даны два списка имён файлов с расширениями. Разработать чистую подпрограмму добавления в первый список имён файлов имён файлов из второго списка таким образом, чтобы в случае нахождения такого файла у него бы менялось расширение на *.bak, а новый добавлялся за ним.

Указание. Элемент списка содержит название файла и его расширение. Применять функцию Index для поиска точки после предварительного прочтения всего имени файла.

§ 5. Лабораторная работа № 3. Однонаправленные и двунаправленные динамические списки

5.1 Общая часть задания

Задание выполняется в виде программного проекта из двух модулей, в котором необходимо использовать *динамические линейные списки*. Списки необходимо обрабатывать чистой хвостовой рекурсией **при всех** операциях с ними. При возможности применяется регулярное программирование.

5.2 Индивидуальные задания

1. В текстовом файле In задан список фамилий (по одной на строке). Разработать процедуры:
   - Формирования линейного однонаправленного динамического списка S с полем строкового типа.
   - Сортировки списка S по алфавиту простым выбором.
   - Вывода в выходной текстовый файл содержимого динамического списка S.
   - Уничтожения динамического списка S.

   С помощью этих процедур отсортировать исходный файл In, записав содержимое отсортированного динамического списка в выходной файл Out и в конце удалив список.
2. В текстовом файле In находятся строки вида:

```
IVANOV 1970
```

Разработать процедуры:

- Создания динамического списка S из записей с полями: строка и целое.
- Вывода динамического списка S в естественном порядке в неформатированный файл из записей Bin.
- Вывода в выходной текстовый файл Out содержимого неформатированного файла записей Bin.
- Удаления динамического списка S.

В головной программе реализовать преобразование и передачу информации по схеме: текстовый файл In → динамический список S → неформатированный файл Bin → выходной текстовый файл Out.

После вывода динамический список S удалить.

Указание. Для формирования неформатированного файла и последующего чтения из него потребуется тип записи *без* поля next. Завести такой тип записи. И только наследованием от него создать тип узла списка с полем next. Разработать: процедуру записи *полей* каждого узла списка в неформатированный файл, процедуру чтения записи (*без* поля next) из созданного неформатированного файла и тут же записи этих данных в форматированный файл.

3. В текстовом файле In задан список фамилий с произвольными номерами вида:

```
11 IVANOV
2  IVANENKO
```

В качестве элемента списка взять запись с полями: целое и строка.

Разработать процедуры:

- Ввода строк записей указанного вида из произвольного текстового файла In в однонаправленный список S.
- Сортировки выбором списка S по возрастанию номеров, стоящих перед фамилией.
- Вывода списка S с указанным типом элемента в текстовый файл Out.
- Уничтожения динамического списка S.

Используя эти процедуры, сформировать файл Out по схеме: In → список → отсортированный список → Out (текстовый).

После вывода динамический список S удалить.

4. В текстовом файле In задан список фамилий с произвольными номерами вида:

```
11 IVANOV
2 IVANENKO и т.д.
```

Разработать процедуры:

- Создания списка S из элементов типа с полями: целое и строка из строк файла In.
- Просмотра элементов списка S и выбрасывания из него узлов, содержащих последнюю заданную фамилию (в том числе и ее саму).
- Вывода содержимого списка S в текстовый файл Out.
- Уничтожения динамического списка S.

Используя эти процедуры, реализовать преобразование и передачу информации по схеме: In → список → преобразованный список без элементов, совпадающих с последним → Out.

После вывода динамический список S удалить.

5. Во входном текстовом файле In находится последовательность строк различной длины. Разработать процедуры:
   - Формирования двунаправленного списка S с элементом типа с полями: строка, next, next_len, где next – ссылка на следующий элемент по порядку в файле, а next_len – пока не реализуемая ссылка.
   - Сортировки списка вставками по возрастанию длины строки (полю next_len).
   - Вывода в текстовый файл Out содержимого двунаправленного списка с элементами указанного вида в обоих направлениях.
   - Уничтожения двунаправленного списка.

   Используя эти процедуры, вывести строки файла In в естественном порядке в файл Out и в отсортированном – в файл Out_len.

   После вывода динамический список S удалить.
6. В текстовом файле In задан список из строк, содержащих фамилию и год рождения:

```
IVANOV 1976
```

   Разработать процедуры:
   - Формирования динамического массива A из строк файла In, используя массив ссылок: элемент массива имеет *производный тип*, содержащий единственное поле-ссылку на *другой производный тип*, содержащий фамилию и год рождения.
   - Сортировки по алфавиту заданного динамического

массива методом выбора.

- Вывода заданного динамического массива в текстовый файл Out.
- Уничтожения динамического массива A.

Используя эти процедуры, отсортировать содержимое файла In по убыванию года рождения и результат вывести в файл Out.

После вывода динамический массив A удалить.

7. В 2-х текстовых файлах In и Delete находятся списки слов (по одному на строке).

Разработать процедуры:

- Формирования линейного списка S с элементом строкового типа из слов заданного текстового файла.
- Выбрасывания из заданного линейного списка S слов заданного файла Delete (если в списке такие слова найдены).
- Вывода линейного списка S в текстовый файл Out.
- Уничтожения динамического списка S.

Используя эти процедуры, выбросить из файла In слова, входящие в файл Delete, выведя результат в файл Out.

После вывода динамический список S удалить.

8. В текстовом файле In задана последовательность символов, в Delete – другая последовательность.

Разработать процедуры:

- Формирования линейного списка S из символов строки текстового файла In.
- Формирования множества M из символов строки заданного текстового файла Delete.
- Исключения из заданного множества M букв ла-

тинского алфавита, входящих в заданный список S.

- Вывода линейного списка символов S в текстовый файл Out.
- Уничтожения динамического списка S.

Множество наследовать от списка. Множество отличается от списка тем, что в нём элементы не повторяются.

Используя эти процедуры, создать список из букв файла In, множество – из букв строки файла Delete, а элементы множества, за исключением входящих в In, вывести в файл Out.

После вывода удалить динамический список S и множество M.

9. В текстовом файле In задан список фамилий (по одной на строке.

Разработать процедуры:

- Создания линейного однонаправленного списка S и записи в него элементов строкового типа.
- Сортировки списка по алфавиту методом Шелла.
- Вывода содержимого списка S в текстовый файл Out.
- Уничтожения динамического списка S.

С помощью этих процедур отсортировать файл In, записав содержимое отсортированного списка S в текстовый файл Out.

После вывода динамический список S удалить.

10. В текстовом файле In задан список фамилий вида:

```
13 SIDOROV
7  POPOV
```

Разработать процедуры:

- Ввода содержимого файла In в список S с элементом с полями: целое и строковое.
- Сортировки вставками списка S по убыванию номеров.
- Вывода списка S в текстовый файл Out.
- Уничтожения динамического списка S.

В головной программе прочитать данные файла In, вывести их, отсортировать их, вывести отсортированные данные.

После вывода динамический список S удалить.

11. В текстовом файле In задан список фамилий (по одной на строке).

Разработать процедуры:

- Формирования линейного списка S из строк текстового файла.
- Удаления из линейного списка S заданной фамилии.
- Вывода содержимого линейного списка S в текстовый файл Out.
- Уничтожения динамического списка S.

В головной программе сделать: все строки за исключением последней в файле In → линейный список; последнюю строку прочитать в отдельную переменную; из линейного списка исключить элементы, совпадающие с этой статической переменной; оставшийся список → Out.

После вывода динамический список S удалить.

12. В текстовом файле In задан список группы по алфавиту с номерами:

```
1  AVERIN
2  BLINOV
```

В файле New_In задан дополнительный список (без номеров и не по алфавиту).

Разработать процедуры:

- Формирования линейного списка S из строк вышеуказанного вида файла In.
- Формирования линейного списка New из строк файла New_In (полиморфный).
- Вставки в однонаправленный список S элементов списка New по алфавиту в нужное место с последующим перенумеровыванием.
- Вывода содержимого линейного списка S в текстовый файл Out.
- Уничтожения динамического списка New (а использовать для S).

Список S наследовать от списка New.

В головной программе создать 2 списка: S и New из строк файлов In и New_In, соответственно, а затем элементы списка New вставить по алфавиту в S и после пронумеровать новый список подряд заново и вывести его.

После вывода динамический список S удалить.

13. В текстовом файле In находится последовательность строк

```
AVERIN VANYA
```

В файле Old находится другой список, в котором могут быть те же лица, что и в Delete.

Разработать процедуры:

- Формирования односвязного списка S из файла In.
- Исключения из заданного списка S элементов файла Delete.
- Вывода содержимого списка S в файл Out.
- Уничтожения динамического списка S.

С помощью этих процедур исключить из списка S (по файлу In) элементы файла Delete. Результат (обновленный список S) вывести в Out, а списки удалить.

14. В текстовом файле In находится список из фамилий с годом рождения через пробелы.

Разработать процедуры:

- Формирования динамического массива S из строк файла In, используя массив ссылок: элемент массива имеет *производный тип*, содержащий единственное поле-ссылку на *другой производный тип*, содержащий фамилию и год рождения.
- Сортировки вставками по возрастанию года рождения заданного динамического массива.
- Вывода заданного динамического массива в текстовый файл Out.
- Уничтожения динамического списка S.

Используя эти процедуры, отсортировать содержимое файла In по возрастанию года рождения и результат вывести в текстовый файл Out.

После вывода динамический список S удалить.

15. В текстовом файле In находится список слов (по одному на строке).

Разработать процедуры:

- Ввода списка слов из текстового файла In в дву-

направленный список S с элементом с полями: строковое, next, next_alph, где next – ссылка на следующий элемент по порядку в файле, а next_alph – пока не реализуемая ссылка.

- Сортировки списка S по алфавиту методом выбора (полю next_alph).
- Вывода содержимого динамического списка по каждому из двух направлений в выходной текстовый файл Out.
- Уничтожения динамического списка S.

В головной программе использовать эти процедуры для сортировки файла In.

После вывода динамический список S удалить.

16. В текстовом файле In задан список фамилий (по одной на строке).

Разработать процедуры:

- Формирования очереди Q с элементом строкового типа.
- Сортировка очереди Q по алфавиту (работа с адресами элементов, см. типичный алгоритм для этого).
- Вывода содержимого очереди Q в выходной текстовый файл Out.
- Уничтожения очереди Q.

С помощью этих процедур отсортировать исходный файл In, записав содержимое отсортированной очереди Q в текстовый файл Out.

После вывода очередь Q удалить.

17. В текстовом файле In задан список фамилий (по одной на строке).

Разработать процедуры:

- Формирования стека S с элементом строкового типа.
- Сортировки стека S по алфавиту (работа с адресами элементов, см. типичный алгоритм для этого).
- Вывода содержимого стека S в выходной текстовый файл Out.
- Уничтожения стека S.

С помощью этих процедур отсортировать исходный файл, записав содержимое отсортированного стека S в текстовый файл Out.

После вывода стек S удалить.

18. В текстовом файле In задан список фамилий (по одной на строке).

Разработать процедуры:

- Формирования однонаправленного списка S с элементом строкового типа.
- Сортировки списка S по алфавиту методом слияния.
- Вывода содержимого линейного списка S в выходной текстовый файл Out.
- Уничтожения динамического списка S.

С помощью этих процедур отсортировать исходный файл In, записав содержимое отсортированного списка S в текстовый файл Out.

После вывода динамический список S удалить.

19. В текстовом файле In задан список фамилий с номерами (не по порядку) вида:

```
11 IVANOV
2  IVANENKO
```

Разработать процедуры:

- Создания стека S из элементов строкового типа из строк файла In.
- Просмотра элементов стека S и выбрасывания из него элементов, содержащих совпадающие с последней записанной в стек фамилией, независимо от их номера, в том числе и последнюю.
- Вывода содержимого стека S в текстовый файл Out.
- Уничтожения стека S.

В головной программе реализовать преобразование и передачу информации по схеме: текстовый файл In → стек → Out; преобразованный стек → Out, используя эти процедуры.

После вывода стек S удалить.

20. В текстовом файле In задан список фамилий (по одной на строке).

Разработать процедуры:

- Формирования очереди Q с элементом строкового типа из строк файла In.
- Уничтожения каждого второго элемента в очереди Q.
- Вывода содержимого очереди Q в естественном порядке в текстовый файл Out.
- Вывода содержимого очереди Q в обратном порядке в текстовый файл Out.

С помощью этих процедур записать содержимое исходного файла In в очередь, вывести очередь в файл Out в прямом и обратном порядке, уничтожить каждый второй элемент очереди Q и снова вывести в прямом и обратном порядке.