

Обработка файловых структур данных

Файловые процедуры и функции

Назначение:

- ☐ организовать доступ к файлам,
 - ☐ осуществить ввод – вывод,
 - ☐ ориентироваться в записях файла,
 - ☐ завершать работу с файлом.
-

файловые переменные

- Для связи Паскаль – программы с внешними устройствами используют файловые переменные
-

Операторы

Связь осуществляется оператором:

□ **ASSIGN (имя файловой переменной,
' имя устройства');**

□ Например:

■ `assign (f, 'book.dat');`

Здесь `f` - имя файловой переменной,

`book.dat` – имя файла данных на внешнем носителе.

Результат: файловая переменная `f`
отожествляется с соответствующим файлом.

Операторы

- Для работы с файлом его необходимо открыть, по окончании работы – закрыть.
 - Файл открывается:
 - для чтения оператор **RESET (f)**
 - для записи оператором **REWRITE (f)**.
 - Файл закрывается:
 - оператор **CLOSE (f)**
-

Чтение и запись данных

- Чтение и запись данных осуществляется известными процедурами read/write, только в начале списка помещается имя файловой переменной:
 - read (f, список ввода);
 - write (f, список вывода);
-

Операторы

Команда

RESET (f)

устанавливает указатель маркера
файла на нулевое состояние,
например, для повторного чтения
записей из файла

Задание

- 1. Создание файла**
 - 2. Обработка файла**
-

Пример 1. Создание файла

- Создать файл, содержащий сведения о студентах
 - Структура записи содержит поля:
 - индекс группы,
 - фамилию студента,
 - курс
 - Количество записей в файле произвольное
-

Пример 2. Обработка файла

- Написать программу определения списка студентов определенного курса с использованием процедуры
-

Создание файла

- Определим поля записи:

Описание полей	Имена переменных	Типы данных	Примеры
Индекс группы	INDEX	символьный, 7 символов	15-ЗИЭ
Фамилия	FAM	символьный, 20 символов	Ильяш А.В.
Курс	KURS	целочисленный	1

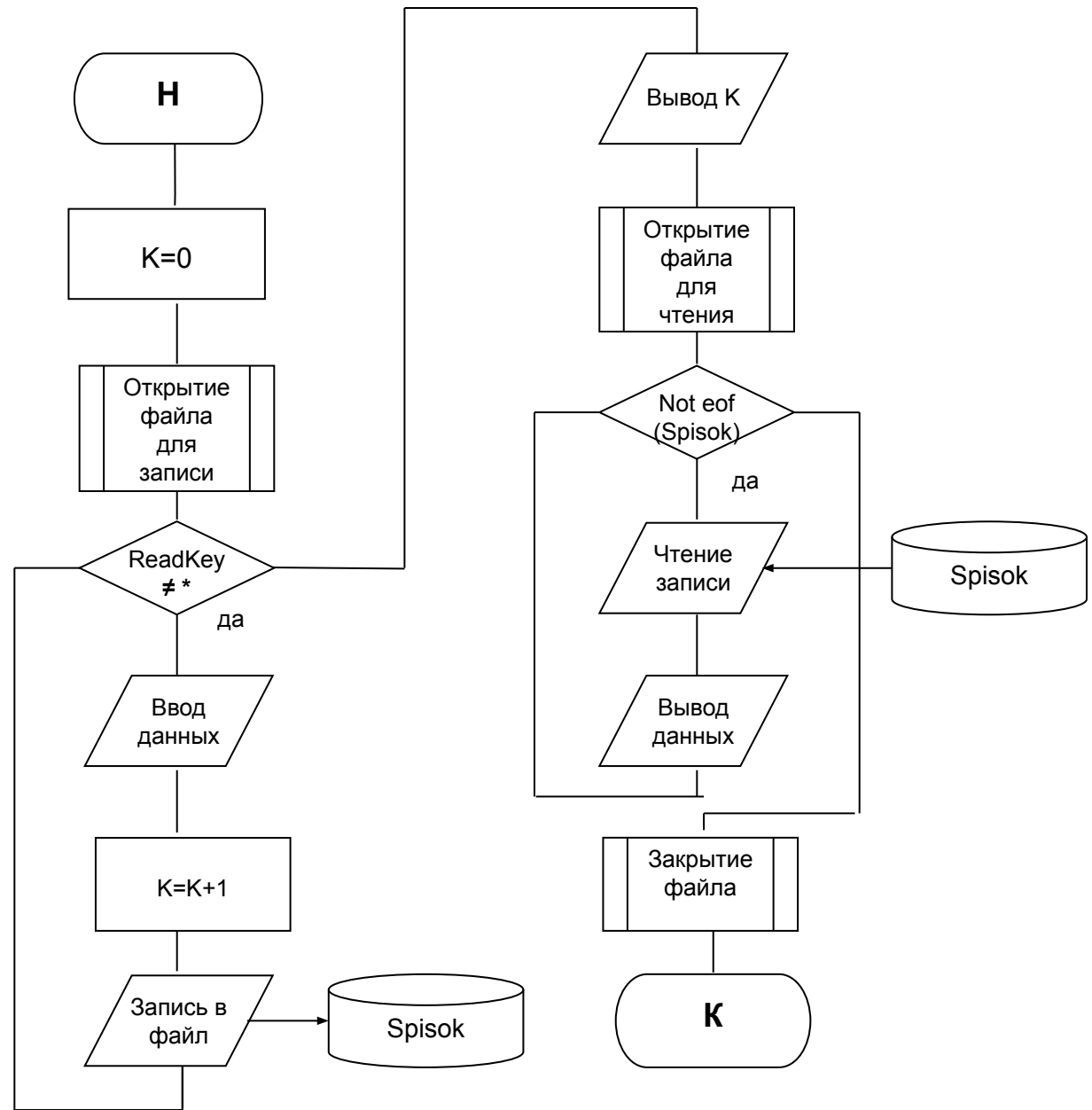
Алгоритм создания файла

- ❑ Для получения текущей записи организуем в программе запрос на ввод очередной порции информации с терминала в оперативную память.
 - ❑ Полученную строку данных запишем в первую запись файла.
 - ❑ Для этого используем оператор записи данных в файл
 - ❑ Затем запросим ввод второй строки данных с терминала в оперативную память.
 - ❑ Организуем ее запись в файл.
 - ❑ Этот процесс ввода с терминала и записи в файл будем продолжать до тех пор, пока не будет получен с терминала признак окончания ввода данных.
 - ❑ Для реализации окончания ввода применим процедуру ReadKey.
-

Алгоритм создания файла

- Для подсчета количества записей в файле введем счетчик K , значение которого будем увеличивать на 1 при каждой операции записи строки в файл.
 - После создания файла на диске для контроля результата организуем чтение записей файла и их вывод на экран
-

Схема алгоритма



Текст программы 1

PROGRAM MY_FILE;

{Алексеев А., 12-ВИЭ-1, вариант 7}

USES CRT;

{Подключение модуля CRT}

{Описание структуры файла}

**TYPE
ZAP=RECORD**

INDEX: STRING[7];

FAM: STRING[20];

KURS: BYTE

END;

VAR

spisok: FILE OF ZAP;

K, L: INTEGER;

X: ZAP;

BEGIN

CLRSCR;

K:=0;

{Связь файловой переменной spisok
с файлом 'spisok' в текущем каталоге}

ASSIGN (spisok, 'spisok');

{Файл открывается для записи}

REWRITE (spisok);

Текст программы 1

{Создание файла}

```
WRITELN ('Для ввода данных нажмите Enter');
WRITELN ('Для окончания работы нажмите *');
  WHILE NOT (READKEY='*') DO
    BEGIN
      WRITELN ('-----');
      WRITE ('Введите индекс группы <=7 символов: ');
      READLN (X.INDEX);
      WRITE ('Введите фамилию <=20 символов: ');
      READLN (X.FAM);
      WRITELN ('Введите курс: ');
      READLN (X.KURS);
      K:=K+1;
      WRITE (x, spisok);
      WRITELN ('Для продолжения ввода нажмите Enter');
      WRITELN ('Для окончания ввода введите *');
    END;
```


Текст программы 1

```
CLRSCR;  
WRITELN ('В файле ',K:5,' записей');  
                                     {Файл открывается для чтения}  
RESET (spisok);  
                                     {Вывод данных файла на экран}  
WHILE NOT EOF (SESSYA) DO  
    BEGIN  
    READ (x,spisok);  
    WRITELN (X.INDEX:8,X.FAM:22, X.KURS:4);  
    END;  
WRITELN ('Конец файла');  
CLOSE (spisok);  
END.
```

Алгоритм обработки файла

Для выполнения задания необходимо:

- в основной программе ввести с терминала переменную KS (курс), по которой определяем список студентов
 - вызвать подпрограмму, определяющую список студентов
 - вывести результаты на экран
-

Алгоритм обработки файла

В подпрограмме:

- организовать в цикле чтение текущей записи файла
 - проверить совпадение значения поля «КУРС» со значением переменной
 - если значения не совпадают, то переходить к чтению следующей записи файла
 - если совпадают, то сохранять фамилию студента и индекс группы, т.е. формировать массив записей
-

Схема
алгоритма

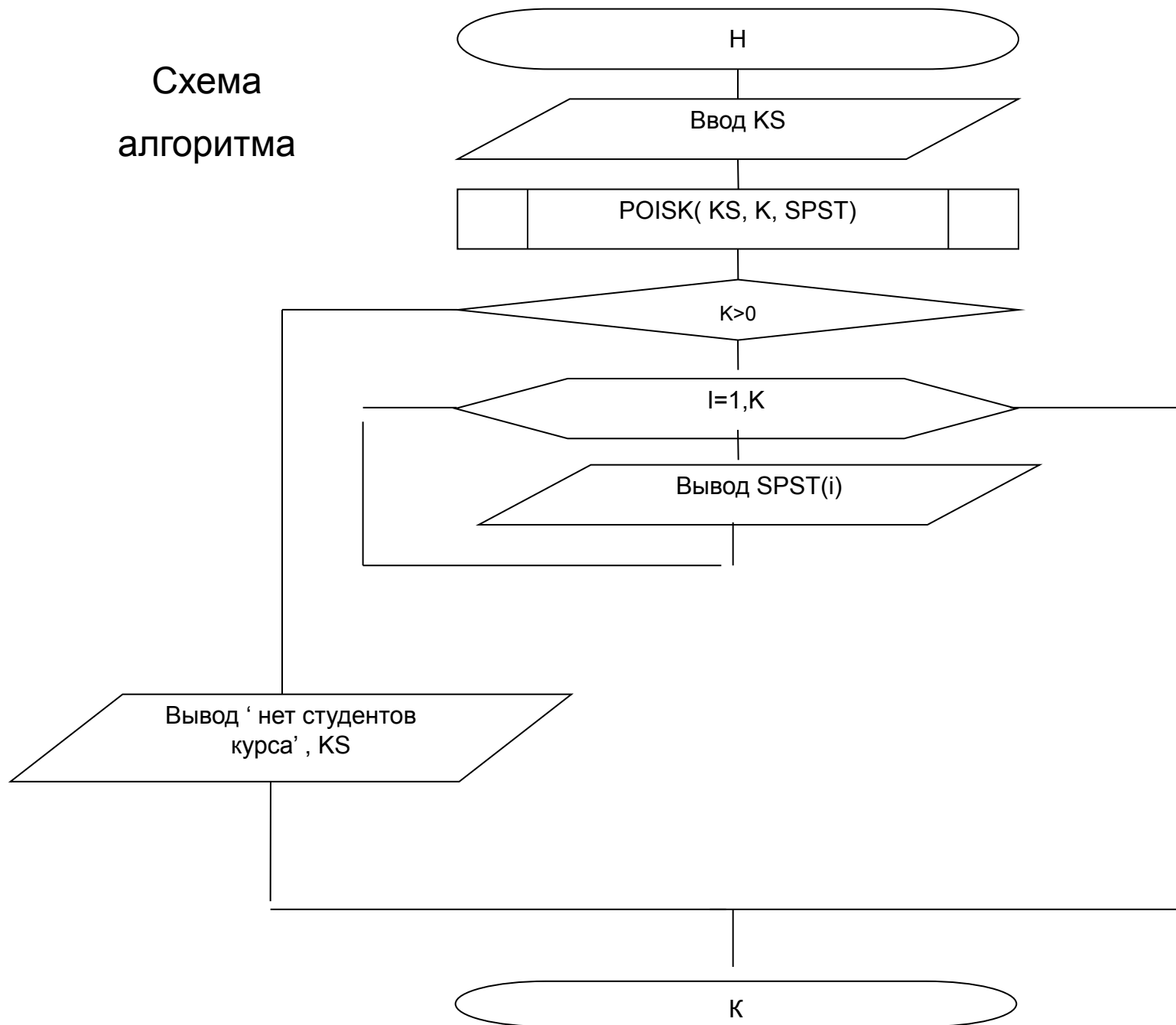
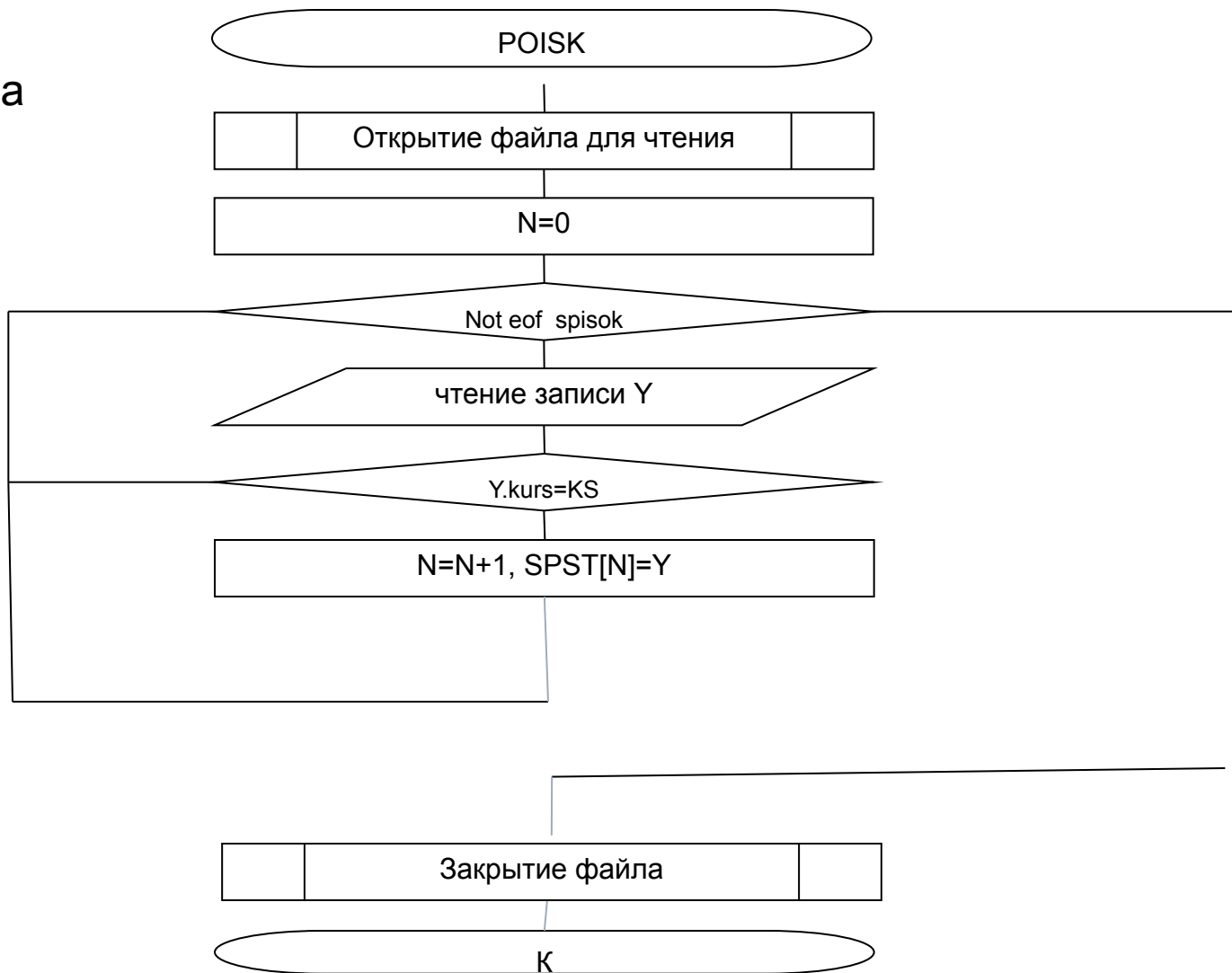


Схема алгоритма



Текст программы обработки

```
Program Spisok_stud;  
{Алексеев А., 15-ЗИЭ-1, вариант 8}  
  Uses Crt;  
    TYPE  
      ZAP=RECORD  
        INDEX: STRING[7];  
        FAM: STRING[20];  
        KURS: BYTE  
      END;  
  Mas= Array [1..20] of Zap;  
  VAR  
    SPST:MAS;  
    KS: BYTE ; I, J, K: Integer;
```

```
Procedure POISK(KS: Byte; Var N: Integer; Var SPST: Mas);  
    Var  
    spisok: File Of Zap;  
    Y: Zap;  
    I: Integer;  
BEGIN  
    ASSIGN (spisok,'spisok');  
    RESET (spisok);  
    N:=0;  
    WHILE NOT EOF (spisok) DO  
        BEGIN  
            READ (Y,spisok);  
            IF Y.Kurs=Ks THEN  
                BEGIN  
                    N:=N+1;  
                    SPST[N]:=Y;  
                END;  
            END;  
        END;  
    CLOSE (spisok);  
END;
```

BEGIN

CLRSCR;

WRITELN ('Введите курс'); READLN (KS);

CLRSCR;

POISK(KS, K, SPST);

IF (K>0) **THEN**

BEGIN

WRITELN ('СПИСОК СТУДЕНТОВ ', KS :3, ' КУРСА');

WRITELN ('_____');

WRITELN (' № Фамилия Группа');

WRITELN ('_____');

FOR I:=1 TO K DO

Begin

WRITE(' | ', I:5, ' | ', SPST[i].FAM :20, ' | ');

WRITE(SPST[I]. INDEX :10);

WRITELN (' | ');

End;

WRITELN ('_____');

END

ELSE WRITELN ('НЕТ СТУДЕНТОВ В ГРУППЕ', KS :3, ' КУРСА');

END.