стр

3. Файловые сортировки

К сожалению, алгоритмы сортировок, приведенные в предыдущем разделе, невозможно применять для данных, которые из-за своего размера не помещаются в оперативной памяти компьютера и находятся на внешних запоминающих устройствах. В таком случае мы говорим, что данные представляют собой *файл данных последовательного доступа*. Это означает, что мы не имеем доступа к произвольному элементу данных, как в массиве, то есть «добраться» до элемента с номером i мы можем, лишь считав последовательно первый элемент, второй элемент, и т.д. до i-го элемента. Более того, для файлов невозможна операция перестановки двух элементов, возможна лишь запись нового элемента в конец файла.

Это весьма сильное ограничение, если сравнивать с возможностями, предоставляемыми массивами, поэтому сортировка файла, в отличие от сортировки массива, всегда требует использования дополнительной памяти (создания дополнительных файлов). При этом приходится пользоваться методами сортировки, основанными на стратегии *слияния*. *Слияние* означает объединение двух (или более) упорядоченных последовательностей в одну упорядоченную последовательность с помощью повторяющегося выбора из доступных на данный момент элементов. Слияние намного проще сортировки, и его используют как вспомогательную операцию в более сложных процессах сортировки последовательностей.

Еще одной особенностью файла является то, что размер файла, в отличие от размера массива, можно считать неограниченным, поэтому при записи элемента в конец файла не надо учитывать количество записанных в файл элементов. Кроме того, число записанных в файл элементов, как правило, неизвестно. Определить, что файл закончился, позволяет специальная функция «конец файла».

стр

3.1. Слияние двух упорядоченных отрезков

Рассмотрим подробнее слияние двух упорядоченных отрезков из файлов B и C. Идея состоит в следующем. Берем с каждого файла по одному элементу. Записываем на файл A меньший из них, и из того файла, в котором находится меньший элемент, читаем следующий элемент. Опять имеем пару элементов с различных файлов. Процесс повторяется до тех пор, пока не закончится один отрезок (какой-то отрезок всегда закончится раньше другого). Затем дописываем оставшийся фрагмент другого отрезка.

Пусть даны два упорядоченных отрезка из файлов B и C, длины которых равны Nb и Nc соответственно. Сольем их в файл A.

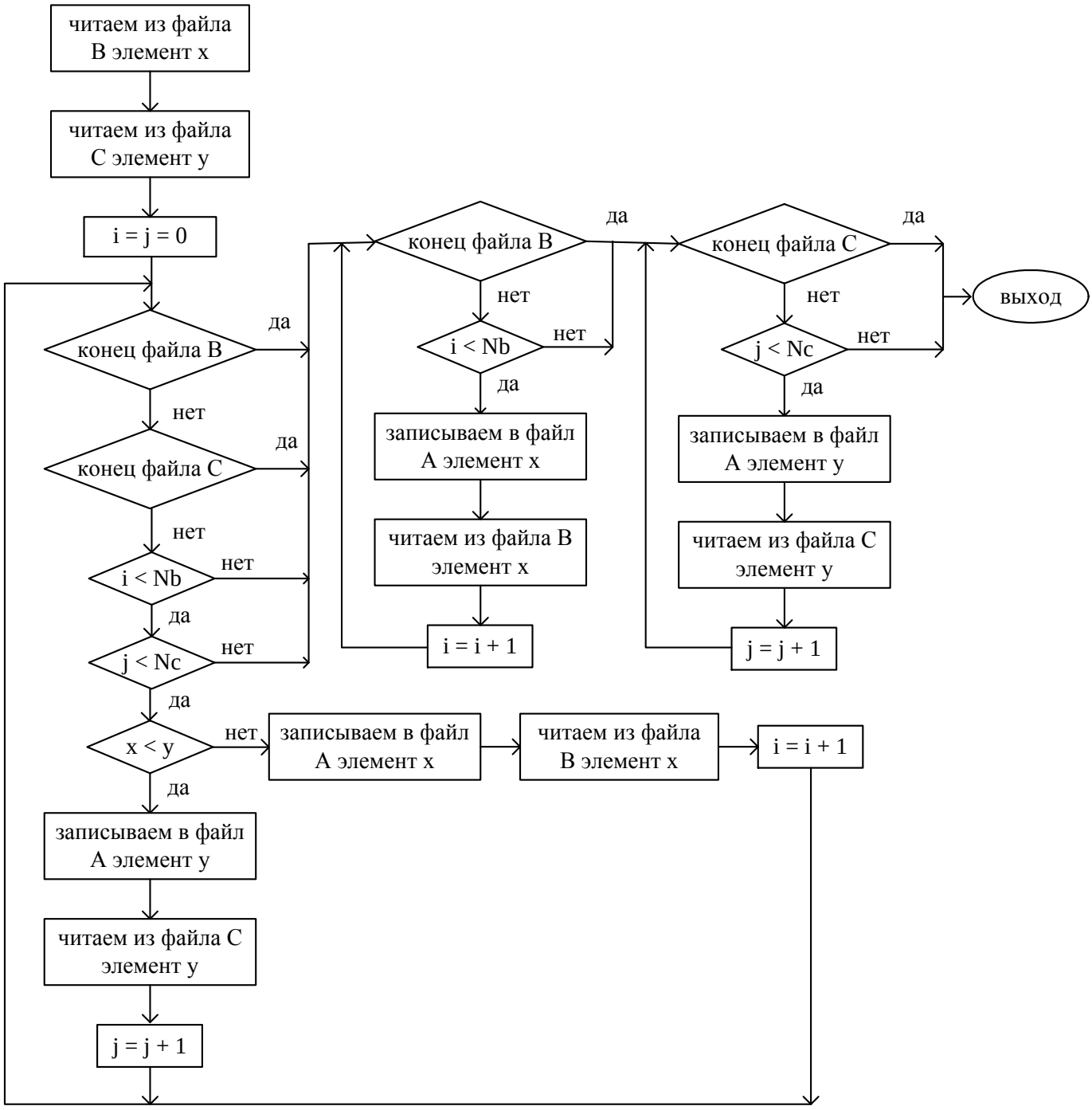
Алгоритм слияния двух упорядоченных отрезков.

1. Читаем из файла B элемент x, а из файла C элемент y.
2. Полагаем $i = j = 0$.
3. Пока не конец файла B и не конец файла C и пока $i < Nb$ и $j < Nc$ выполняем:
   - 3.1. если $x < y$, то делаем:
     - 3.1.1. записываем в файл A элемент x;
     - 3.1.2. читаем из файла B элемент x;
     - 3.1.3. $i = i + 1$;
   - 3.2. иначе:
     - 3.2.1. записываем в файл A элемент y;
     - 3.2.2. читаем из файла C элемент y;
     - 3.2.3. $j = j + 1$.
4. Пока не конец файла B и $i < Nb$ выполняем:

4.1. записываем в файл *A* элемент *x*;
4.2. читаем из файла *B* элемент *x*;
4.3. *i* = *i* + 1.

5. Пока не конец файла *C* и *j* < *Nc* выполняем:
5.1. записываем в файл *A* элемент *y*;
5.2. читаем из файла *C* элемент *y*;
5.3. *j* = *j* + 1.

Блок-схема слияния двух упорядоченных отрезков.



стр

3.2. Прямое слияние

3.2.1. Прямое слияние на трех файлах

Одна из сортировок на основе слияния называется *прямым слиянием на трех файлах*. Она выполняется следующим образом:

1. Последовательность *A* разбивается на две половины.
2. Части *B* и *C* сливаются в последовательность *A*, при этом одиночные элементы образуют упорядоченные пары.
3. Полученная последовательность *A* вновь обрабатывается как указано в пунктах 1, 2, при этом упорядоченные пары переходят в упорядоченные четверки.
4. Повторяя предыдущие шаги, сливаем четверки в восьмерки и так далее, каждый раз «удваивая» длину слитых упорядоченных подпоследовательностей до тех пор, пока не будет упорядочена вся последовательность.

Рассмотрим алгоритм на примере. Последовательность *A* представлена ниже, подпоследовательности *B* и *C* выделены разными цветами. Механизм разбиения таков: первый элемент записываем в *B*, второй – в *C*, третий – в *B*, четвертый – в *C*, и так далее до конца последовательности *A*.

A: 21 12 14 34 18 53 29 36 48 19 43

Разбиваем последовательность *A* на две половины: *B* и *C*.

B: 21 14 18 29 48 43
C: 12 34 53 36 19

Теперь сливаем полученные последовательности в последовательность *A*, упорядочивая при этом пары элементов (расположенные в столбцах и выделенные разными цветами). Получаем последовательность из упорядоченных пар. Так как число элементов в файле нечетное, последний элемент остался без пары.

A: 12 21 14 34 18 53 29 36 19 48 43

Опять разбиваем последовательность *A* на две половины: *B* и *C*, которые в *A* выделены разными цветами. Первую упорядоченную пару записываем в *B*, вторую – в *C*, третью – в *B*, четвертую – в *C*, и так далее до конца последовательности *A*.

A: 12 21 14 34 18 53 29 36 19 48 43

B: 12 21 18 53 19 48
C: 14 34 29 36 43

стр

Теперь сливаем подпоследовательности в последовательность *A*, объединяя выделенные разными цветами упорядоченные пары в упорядоченные четверки элементов.

A: 12 14 21 34 18 29 36 53 19 43 48

Опять разбиваем последовательность *A* на две половины, теперь уже записывая первую четверку в *B*, вторую – в *C*, и так далее до конца файла.

A: 12 14 21 34 18 29 36 53 19 43 48

B: 12 14 21 34 19 43 48

C: 18 29 36 53

Сливаем подпоследовательности, объединяя упорядоченные четверки в упорядоченные восьмерки.

A: 12 14 18 21 29 34 36 53 19 43 48

Опять разбиваем последовательность на две половины, теперь уже по восьмеркам (последняя восьмерка неполная).

A: 12 14 18 21 29 34 36 53 19 43 48

B: 12 14 18 21 29 34 36 53
C: 19 43 48

В последний раз выполняем слияние подпоследовательностей (так как каждая из них содержит по одной упорядоченной восьмерке). Получаем упорядоченную последовательность.

A: 12 14 18 19 21 29 34 36 43 48 53

Действия по однократной обработке всего множества данных называются *фазой*. Наименьший же подпроцесс, повторение которого составляет процесс сортировки, называется проходом или этапом. В приведенном выше примере сортировка происходит за три прохода, каждый из которых состоит из фазы разделения и фазы слияния. Для выполнения такой сортировки нужны три файла, поэтому она называется *прямым слиянием на трех файлах*.

стр

3.2.1.1. Фаза разбиения

Рассмотрим подробнее фазу разделения последовательности *A* на две последовательности *B* и *C*. Идея состоит в следующем. Берем *p* элементов из последовательности *A* и переписываем их в последовательность *B*, затем снова берем *p* элементов из последовательности *A* и переписываем их в последовательность *C*. Снова переписываем отрезок длины *p* в последовательность *B* и *C* по очереди. И так продолжаем до тех пор, пока не закончится последовательность *A*.

Пусть дана последовательность *A*. Разобьем ее на две последовательности *B* и *C* отрезками длины *p*.

Приведем пример разбиения файла *A* при $p = 4$. Текущий считанный элемент файла выделен в рамку, индекс *j* содержит число элементов текущего отрезка, уже переписанных в файл *B* или *C*.

i = 0
A: [12] 14 21 34 18 29 36 53 19 43 48

Записываем текущий элемент файла *A* в файл *B*, увеличивая индекс *i* на единицу. Повторяем этот шаг до тех пор, пока $i < p$.

B: 12

C:

---

i = 1

A: 12 [14] 21 34 18 29 36 53 19 43 48

B: 12 14

C:

---

i = 2

A: 12 14 [21] 34 18 29 36 53 19 43 48

B: 12 14 21

C:

---

i = 3

A: 12 14 21 [34] 18 29 36 53 19 43 48

B: 12 14 21 34

C:

---

i = 4

A: 12 14 21 34 [18] 29 36 53 19 43 48

стр

Здесь *i* = 4, это означает, что первый отрезок закончился. Полагаем *i* = 0, и переписываем следующий отрезок в файл *C*.

i = 0

A: 12 14 21 34 [18] 29 36 53 19 43 48

B: 12 14 21 34

C: 18

---

i = 1

A: 12 14 21 34 18 [29] 36 53 19 43 48

B: 12 14 21 34

C: 18 29

---

i = 2

A: 12 14 21 34 18 29 [36] 53 19 43 48

B: 12 14 21 34

C: 18 29 36

---

i = 3

A: 12 14 21 34 18 29 36 [53] 19 43 48

B: 12 14 21 34

C: 18 29 36 53

---

i = 4

A: 12 14 21 34 18 29 36 53 [19] 43 48

стр

Здесь *i* = 4, это означает, что второй отрезок закончился. Полагаем *i* = 0, и переписываем следующий отрезок в файл *B*.

---

i = 0

A: 12 14 21 34 18 29 36 53 [19] 43 48

B: 12 14 21 34 19

C: 18 29 36 53

---

i = 1

A: 12 14 21 34 18 29 36 53 19 [43] 48

B: 12 14 21 34 19 43

C: 18 29 36 53

---

i = 2

A: 12 14 21 34 18 29 36 53 19 43 [48]

B: 12 14 21 34 19 43 48

C: 18 29 36 53

---

i = 3

A: 12 14 21 34 18 29 36 53 19 43 48

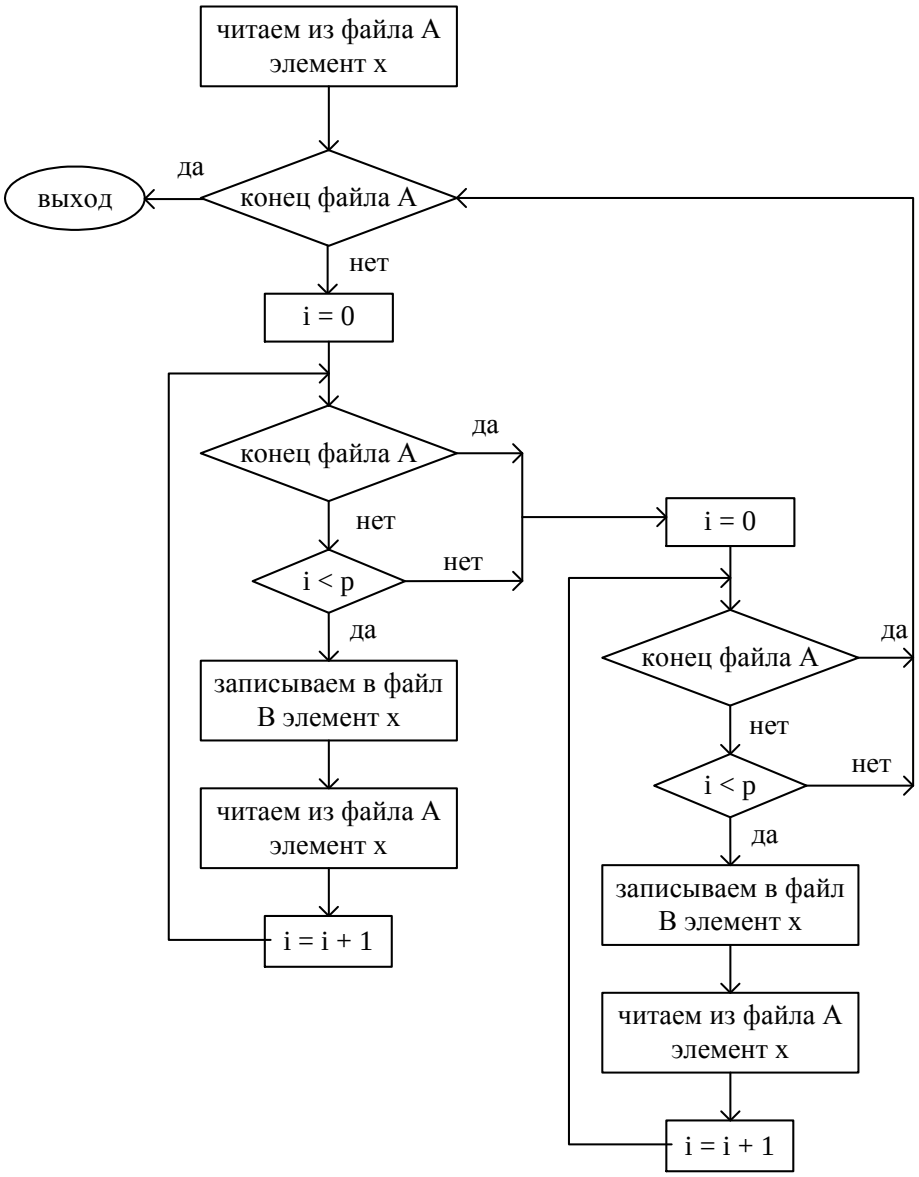
Достигнут конец файла *A*, значит, работа алгоритма завершена.

стр

Алгоритм фазы разбиения

1. читаем из файла *A* элемент x;
2. пока не конец файла *A* выполняем:
   2.1. *i* = 0;
   2.2. пока не конец файла *A* и пока i < *p* выполняем:
      2.2.1. записываем в файл *B* элемент *x*;
      2.2.2. читаем из файла *A* элемент *x*;
      2.2.3. *i* = *i* + 1;
   2.3. *i* = 0;
   2.4. пока не конец файла *A* и *i* < *p* выполняем:
      2.4.1. записываем в файл *C* элемент *x*;
      2.4.2. читаем из файла *A* элемент *x*;
      2.4.3. *i* = *i* + 1;

Блок-схема фазы разбиения



стр

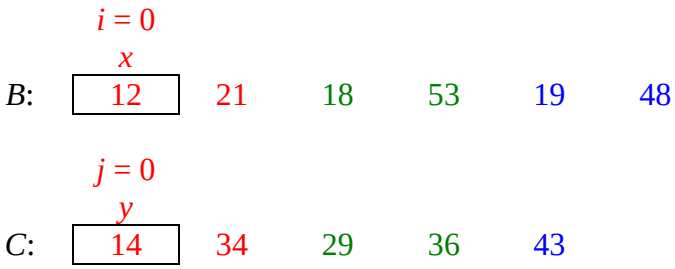
3.2.1.2. Фаза слияния

Теперь рассмотрим подробнее фазу слияния двух частично упорядоченных файлов *B* и *C*. Идея состоит в следующем. Берем с каждого файла по одному упорядоченному *отрезку* длины *p*. Сливаем два упорядоченных файла в файл *A*. Опять берем с каждого файла по одному упорядоченному *отрезку* длины *p* и сливаем их в файл *A*. Процесс повторяется до тех пор, пока не закончится один файл. Затем дописывает оставшийся фрагмент с оставшегося файла.

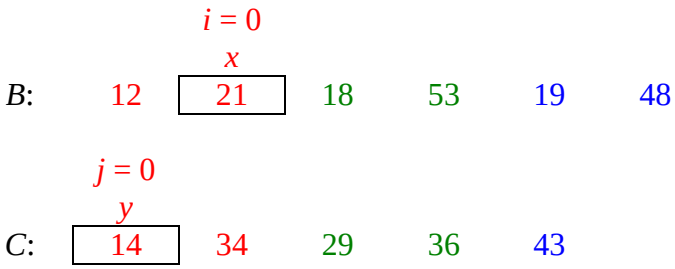
Пусть даны два частично упорядоченных файла *B* и *C*, длина упорядоченного отрезка *p*=2. Сольем их в файл *A*.

B:	12	21	18	53	19	48
C:	14	34	29	36	43	

Будем считывать элементы из файла *B* в переменную *x*, элементы из файла *C* в переменную *y*. Последние считанные из файлов элементы обведены в рамку. Индексы *i* и *j* соответственно означают число уже записанных в файл *A* элементов текущих отрезков из файлов *B* и *C*.

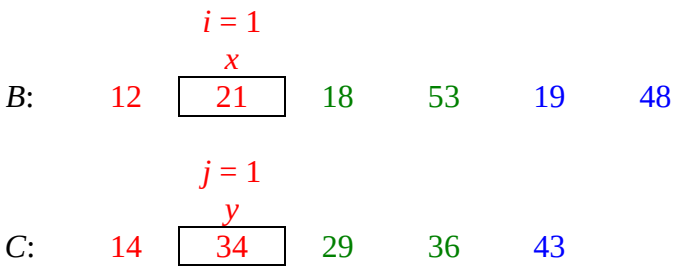


Здесь $x < y$, поэтому записываем элемент $x = 12$ в файл A, считываем следующий элемент из файла B и увеличиваем индекс i на единицу.

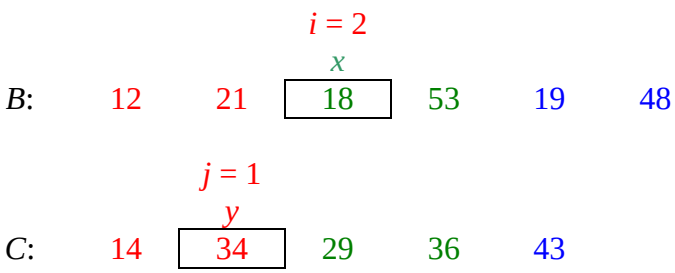


стр

Здесь $y < x$, поэтому записываем элемент $y = 14$ в файл A, считываем следующий элемент из файла C и увеличиваем индекс j на единицу.



Здесь $x < y$, поэтому записываем элемент $x = 21$ в файл A, считываем следующий элемент из файла B и увеличиваем i на единицу.



Индекс $i = 2 = p$, это означает, что достигнут конец упорядоченного отрезка из файла *B*. Поэтому дописываем конец упорядоченного отрезка из файла *B* в файл *A*.

A: 12 14 21 34

$i = 2$
x
B: 12 21 [18] 53 19 48

$j = 2$
y
C: 14 34 [29] 36 43

Здесь индексы $i = j = 2 = p$, это означает, что упорядоченные отрезки в обоих файлах закончились. Так как при этом оба файла еще не закончились, полагаем $i = j = 0$, т. е. начинаем слияние двух следующих отрезков.
стр

$i = 0$
x
B: 12 21 [18] 53 19 48

$j = 0$
y
C: 14 34 [29] 36 43

Здесь $x < y$, поэтому записываем элемент $x = 18$ в файл *A*, считываем следующий элемент из файла *B* и увеличиваем *i* на единицу.

A: 12 14 21 34 18

$i = 1$
x
B: 12 21 18 [53] 19 48

$j = 0$
y
C: 14 34 [29] 36 43

Здесь $y < x$, поэтому записываем элемент $y = 29$ в файл *A*, считываем следующий элемент из файла *C* и увеличиваем индекс *j* на единицу.

A: 12 14 21 34 18 29

					$i = 1$		
					x		
B:	12	21	18	53	19	48	
					$j = 1$		
					y		
C:	14	34	29	36	43		

Здесь снова $y < x$, поэтому записываем элемент $y = 36$ в файл A, считываем следующий элемент из файла C и увеличиваем индекс j на единицу.

A:	12	14	21	34	18	29	36

					$i = 0$		
					x		
B:	12	21	18	53	19	48	
						$j = 2$	
						y	
C:	14	34	29	36	43		

Индекс $j = 2 = p$, это означает, что достигнут конец упорядоченного отрезка из файла C. Поэтому дописываем конец упорядоченного отрезка из файла C в файл A.

A:	12	14	21	34	18	29	36	53

						$i = 2$	
						x	
B:	12	21	18	53	19	48	
						$j = 2$	
						y	
C:	14	34	29	36	43		

стр

Здесь индексы $i = j = 2 = p$, это означает, что упорядоченные отрезки в обоих файлах закончились. Так как при этом оба файла еще не закончились, полагаем $i = j = 0$, т. е. начинаем слияние двух следующих отрезков.

						$i = 0$	
						x	
B:	12	21	18	53	19	48	
						$j = 0$	
						y	
C:	14	34	29	36	43		

Здесь $x < y$, поэтому записываем элемент $x = 19$ в файл A, считываем следующий элемент из файла B и увеличиваем i на единицу.

A: 12 14 21 34 18 29 36 53 19

i = 1
x

B: 12 21 18 53 19 | 48 |

j = 0
y

C: 14 34 29 36 | 43 |

Здесь *y* < *x*, поэтому записываем элемент *y* = 43 в файл *A*, считываем следующий элемент из файла *C* и увеличиваем индекс *j* на единицу.

A: 12 14 21 34 18 29 36 53 19 43

i = 1
x

B: 12 21 18 53 19 | 48 |

j = 1
y

C: 14 34 29 36 43 | |

Файл *C* закончился, поэтому дописываем остаток файла *B* в файл *A*.

A: 12 14 21 34 18 29 36 53 19 43 48

i = 0
x

B: 12 21 18 53 19 48 | |

j = 0
y

C: 14 34 29 36 43 | |

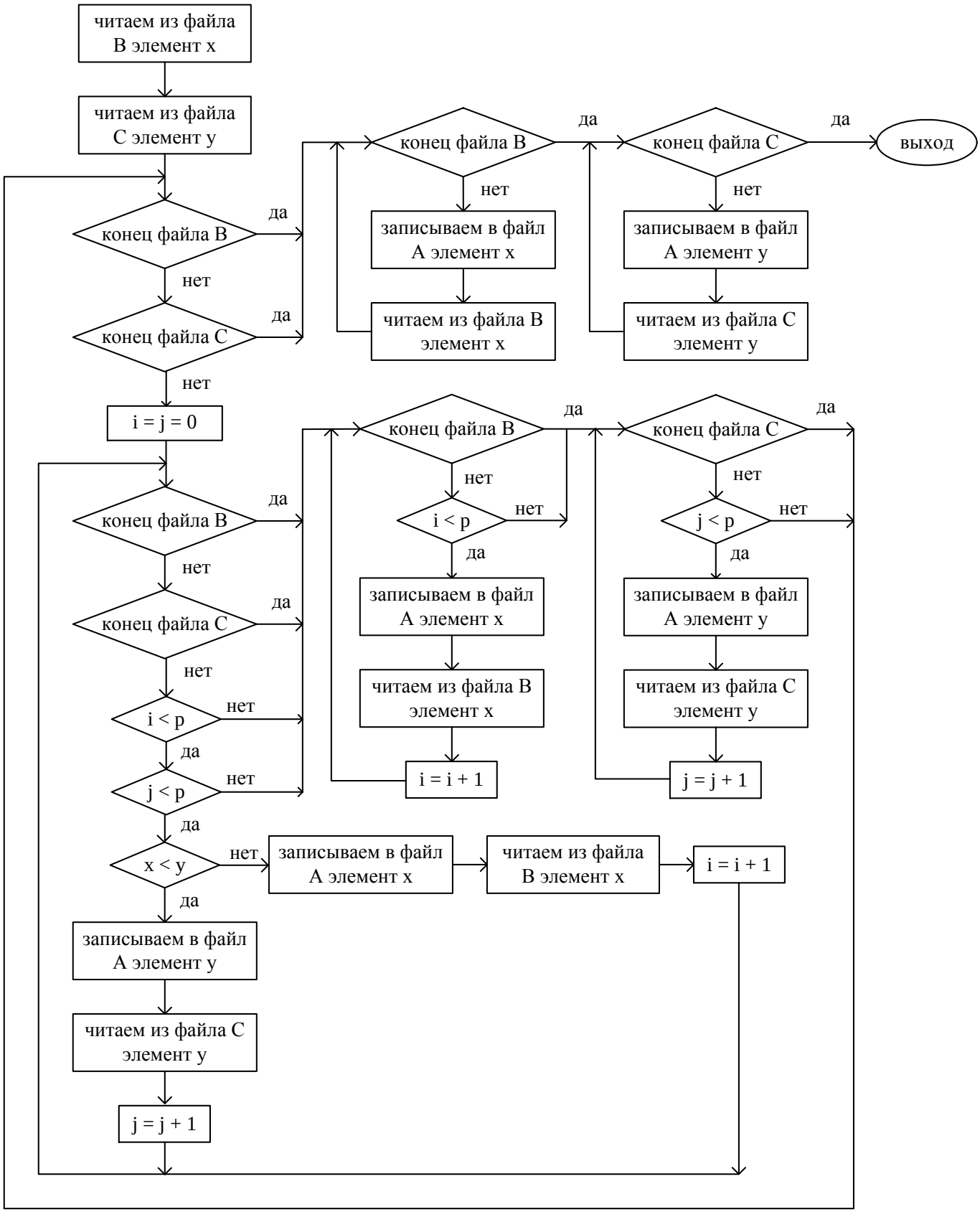
Файлы *B* и *C* закончились, значит, процесс слияния файлов завершен.
стр

Алгоритм фазы слияния

1. Читаем из файла *B* элемент *x*, а из файла *C* элемент *y*.
2. Пока не конец файла *B* и не конец файла *C* выполняем:
   2.1. *i* = *j* = 0;
   2.2. пока не конец файла *B* и не конец файла *C* и пока i < *p* и j < *p* выполняем:
      2.2.1. если *x* < *y*, то делаем:
         2.2.1.1 записываем в файл *A* элемент *x*;

2.2.1.2. читаем из файла *B* элемент *x*;
2.2.1.3. $i = i + 1$;
2.2.2. иначе:
2.2.2.1. записываем в файл *A* элемент *y*;
2.2.2.2. читаем из файла *C* элемент *y*;
2.2.2.3. $j = j + 1$;
2.3. пока не конец файла *B* и $i < p$ выполняем:
2.3.1. записываем в файл *A* элемент *x*;
2.3.2. читаем из файла *B* элемент *x*;
2.3.3. $i = i + 1$;
2.4. пока не конец файла *C* и $j < p$ выполняем:
2.4.1. записываем в файл *A* элемент *y*;
2.4.2. читаем из файла *C* элемент *y*;
2.4.3. $j = j + 1$.
3. Пока не конец файла *B* выполняем:
3.1. записываем в файл *A* элемент *x*;
3.2. читаем из файла *B* элемент *x*.
4. Пока не конец файла *C* выполняем:
4.1. записываем в файл *A* элемент *y*;
4.2. читаем из файла *C* элемент *y*.

Блок-схема фазы слияния.



стр

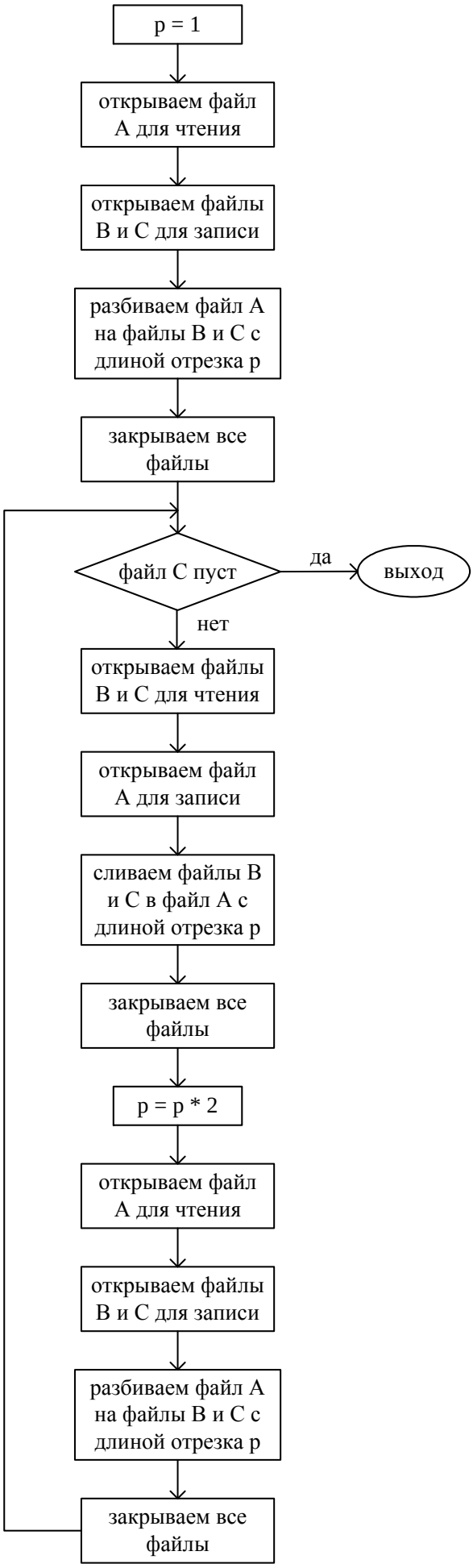
3.2.1.3. Сортировка прямым слиянием на трех файлах

И, наконец, собственно сортировка прямым слиянием на трех лентах состоит из следующих шагов.

Алгоритм сортировки прямым слиянием на трех файлах.

1. Полагаем $p = 1$.
2. Открываем файл *A* для чтения.
3. Открываем файлы *B* и *C* для записи.
4. Разбиваем файл *A* на файлы *B* и *C* с длиной отрезка *p*.
5. Закрываем все файлы.
6. Если файл *C* не пуст выполняем:
7. открываем файлы *B* и *C* для чтения;
8. открываем файл *A* для записи;
9. сливаем файлы *B* и *C* в файл *A* с длиной отрезка *p*;.
10. закрываем все файлы;
11. полагаем $p = 2\ p$;
12. открываем файл *A* для чтения;
13. открываем файлы *B* и *C* для записи;
14. разбиваем файл *A* на файлы *B* и *C* с длиной отрезка *p*;
15. закрываем все файлы.
16. Повторяем шаги с 6 по 16.

Блок-схема сортировки прямым слиянием на трех файлах.


p = 1
открываем файл A для чтения
открываем файлы B и C для записи
разбиваем файл A на файлы B и C с длиной отрезка p
закрываем все файлы
файл C пуст
да
выход
нет
открываем файлы B и C для чтения
открываем файл A для записи
сливаем файлы B и C в файл A с длиной отрезка p
закрываем все файлы
p = p * 2
открываем файл A для чтения
открываем файлы B и C для записи
разбиваем файл A на файлы B и C с длиной отрезка p
закрываем все файлы

Пример (ссылка) сортировки прямым слиянием на трех файлах был приведен выше.
стр

3.2.2. Прямое слияние на четырех файлах

Фазы разделения (ссылка) фактически не относятся к сортировке, ведь в них элементы не переставляются. В некотором смысле они не продуктивны, хотя и занимают половину всех операций по переписи. Если объединить разделение со слиянием, то от этих переписей вообще можно избавиться. Вместо слияния в одну последовательность результаты слияния будем сразу распределять по двум файлам, которые станут исходными для последующего прохода. В отличии от предыдущей *двухфазной* сортировки с помощью слияния будем называть такую сортировку *однофазной*. Она очевидно лучше, но за это приходится «платить» четвертым файлом.

Рассмотрим алгоритм на примере. Сначала разобьем файл *A* на файлы *C* и *D*. Последовательность *A* представлена ниже, подпоследовательности *C* и *D* выделены разными цветами. Механизм разбиения такой же, как и в сортировке на трех файлах: первый элемент записываем в *C*, второй – в *D*, третий – в *C*, четвертый – в *D*, и так далее до конца последовательности *A*.

A:	21	12	14	34	18	53	29	36	48	19	43

C:	21	14	18	29	48	43
D:	12	34	53	36	19	

Теперь сливаем полученные последовательности в файлы *A* и *B*, упорядочивая при этом пары элементов (расположенные в столбцах и выделенные разными цветами). Первую пару записываем в файл *A*, вторую – в файл *B*, третью – снова в файл *A*, четвертую – в файл *B*, и т.д. до конца файлов *C* и *D*. Получаем две последовательности из упорядоченных пар. Так как число элементов в файле нечетное, последний элемент остался без пары.

A:	12	21	18	53	19	48
B:	14	34	29	36	43	

стр

Аналогично сливаем подпоследовательности в файлы *C* и *D*, объединяя выделенные разными цветами упорядоченные пары в упорядоченные четверки элементов (последняя четверка неполная).

A:	12	21	18	53	19	48
B:	14	34	29	36	43	

C:	12	14	21	34	19	43	48
D:	18	29	36	53			

Сливаем подпоследовательности в файлы *A* и *B*, объединяя упорядоченные четверки в упорядоченные восьмерки (последняя восьмерка неполная).

C:	12	14	21	34	19	43	48
D:	18	29	36	53			

A:	12	14	18	21	29	34	36	53

B: 19 43 48

Аналогично сливаем подпоследовательности в файлы *C* и *D*, объединяя выделенные разными цветами упорядоченные восьмерки.

A: 12 14 18 21 29 34 36 53
B: 19 43 48

C: 12 14 18 19 21 29 34 36 43 48 53
D:

Файл *D* оказался пустым, значит, сортировка закончена. Упорядоченная последовательность находится в файле *C*.

Теперь перейдем к более детальному рассмотрению алгоритма сортировки файлов прямым слиянием на четырех файлах.

стр

3.2.2.1. Фаза разбиения на четырех файлах

На самом деле фаза разбиения сортировки прямым слиянием на четырех файлах ничем не отличается от фазы разбиения прямым слиянием на трех файлах (ссылка). Но, как вы могли заметить, в блок-схеме и в словесном описании алгоритма повторяются одинаковые части, отличающиеся лишь именем файла. Чтобы избежать повторов и сделать описание более компактным, мы создадим массив, состоящий из файлов, куда будет вестись запись упорядоченных отрезков длины *p*.

Нам потребуется следующие переменные:

x – значение текущего элемента из файла *A*;
F[2] – массив из двух файлов, на которые разбивается файл *A*;
n – номер файла, в который записывается текущий отрезок.

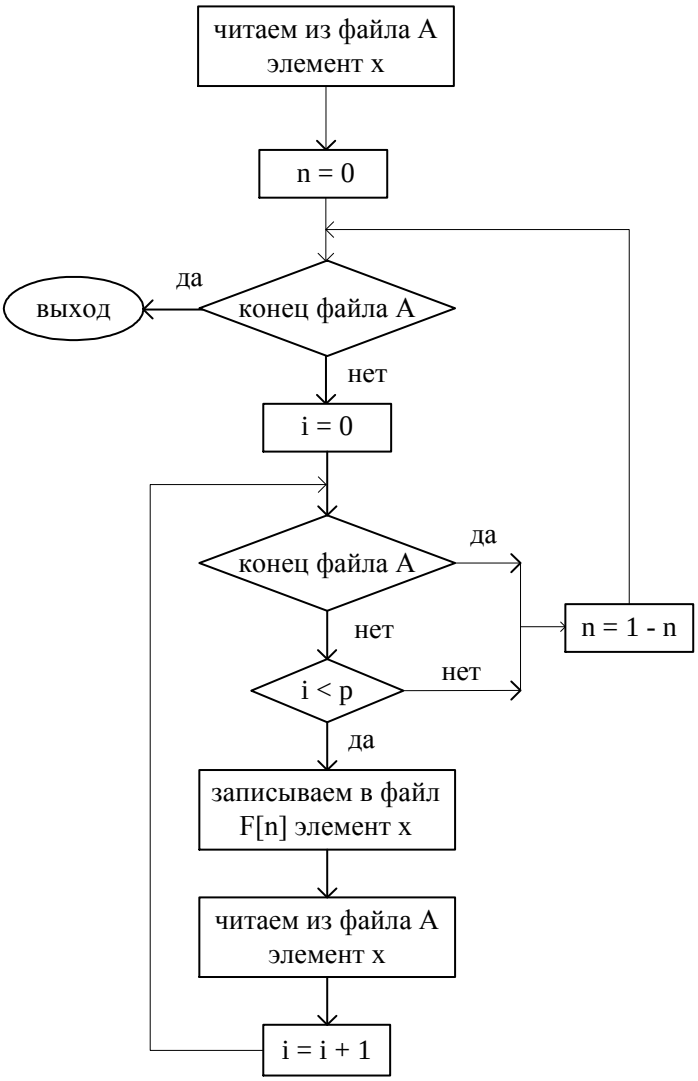
Индекс *n* меняется с 0 на 1 и с 1 на 0, т.е. для его изменения нужно положить $n = 1 - n$. Изменение индекса происходит каждый раз, когда заканчивается очередной упорядоченный отрезок длины *p*, на данном этапе равном 1.

Алгоритм фазы разбиения на четырех файлах

1. Читаем из файла *A* элемент *x*.
2. Полагаем $n = 0$.
3. пока не конец файла *A* выполняем:
   3.1. полагаем $i = 0$;
   3.2. пока не конец файла *A* и пока $i < p$ выполняем:
      3.2.1. записываем в файл *F*[*n*] элемент *x*;
      2.2.2. читаем из файла *A* элемент *x*;
      2.2.3. полагаем $i = i + 1$;
   3.3. полагаем $n = 1 - n$.

Конечно, этот вариант алгоритма можно использовать и в прямом слиянии на трех файлах.

Блок-схема фазы разбиения на четырех файлах



стр

3.2.2.2. Фаза слияния на четырех файлах

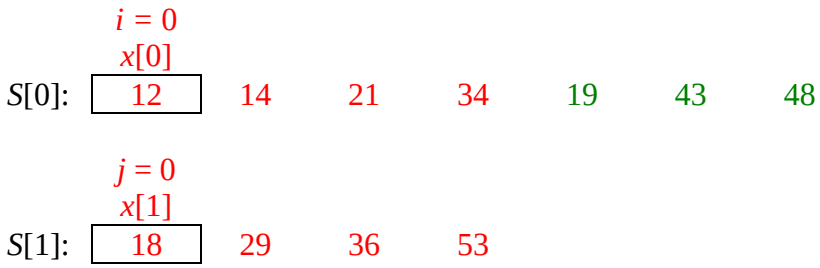
Сама идея фазы слияния на четырех файлах не отличается от идеи фазы слияния на трех файлах. Но, в отличии от трех файлов, при слиянии на четырех файлах само слияние проводится не в один файл, а в два попеременно. Поэтому здесь удобно использовать не отдельные файлы, а два массива по два файла.

S[2] – массив из двух файлов, из которых производится чтение упорядоченных отрезков;
F[2] – массив из двух файлов, в которые производится запись объединенных упорядоченных отрезков;
x[2] – массив из текущих элементов, считанных из файлов *S*[0], *S*[1];
n – номер файла, в который записывается текущий отрезок;
i – длина текущего отрезка из файла *S*[0];
j - длина текущего отрезка из файла *S*[1];
p - длина отрезка на данном этапе.

Пусть даны два частично упорядоченных файла *S*[0] и *S*[1], длина упорядоченного отрезка *p*=4. Сольем их в файлы *F*[0] и *F*[1].

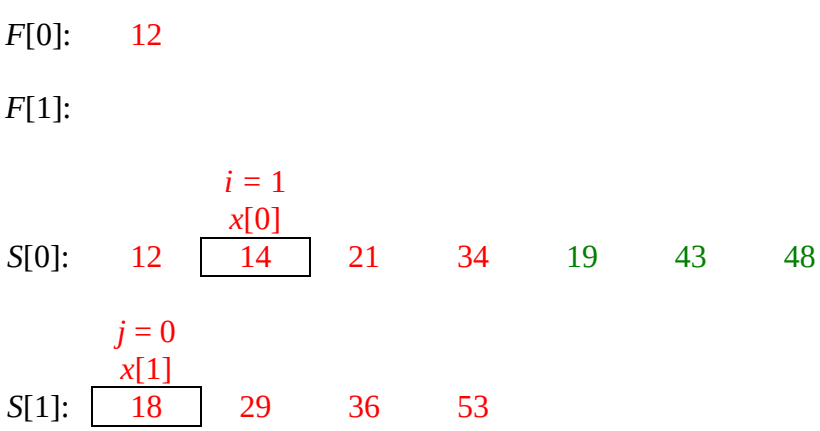
S[0]:	12	14	21	34	19	43	48
S[1]:	18	29	36	53			

Будем считывать элементы из файла $S[0]$ в переменную $x[0]$, элементы из файла $S[1]$ в переменную $x[1]$. Последние считанные из файлов элементы обведены в рамку. Индексы i и j соответственно означают число уже записанных в файлы $F[0]$ и $F[1]$ элементов текущих отрезков из файлов $S[0]$ и $S[1]$. Полагаем $n = 0$, т.е. будем записывать отрезок в файл $F[0]$.

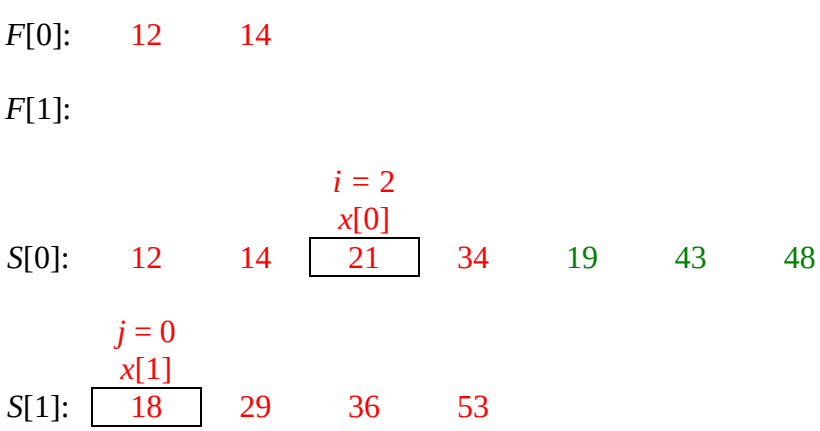


стр

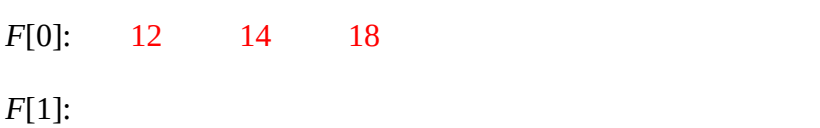
Здесь $x[0] < x[1]$, поэтому записываем элемент $x[0] = 12$ в файл $F[n] = F[0]$. Увеличиваем индекс i на единицу и считываем из файла $S[0]$ очередной элемент.



Здесь $x[0] < x[1]$, поэтому записываем элемент $x[0] = 14$ в файл $F[n] = F[0]$. Увеличиваем индекс i на единицу и считываем из файла $S[0]$ очередной элемент.



Здесь $x[1] < x[0]$, поэтому записываем элемент $x[1] = 18$ в файл $F[n] = F[0]$. Увеличиваем индекс j на единицу и считываем из файла $S[1]$ очередной элемент.



			$i = 2$ $x[0]$				
$S[0]$:	12	14	21	34	19	43	48
		$j = 1$ $x[1]$					
$S[1]$:	18	29	36	53			

стр

Действуя аналогично, получаем:

$F[0]$:	12	14	18				
$F[1]$:							
			$i = 2$ $x[0]$				
$S[0]$:	12	14	21	34	19	43	48
		$j = 1$ $x[1]$					
$S[1]$:	18	29	36	53			

$F[0]$:	12	14	18	21			
$F[1]$:							
				$i = 3$ $x[0]$			
$S[0]$:	12	14	21	34	19	43	48
		$j = 1$ $x[1]$					
$S[1]$:	18	29	36	53			

$F[0]$:	12	14	18	21	29		
$F[1]$:							
				$i = 3$ $x[0]$			
$S[0]$:	12	14	21	34	19	43	48
			$j = 2$ $x[1]$				
$S[1]$:	18	29	36	53			

F[0]: 12 14 18 21 29 34

F[1]:

i = 4
x[0]
S[0]: 12 14 21 34 [19] 43 48

j = 2
x[1]
S[1]: 18 29 [36] 53

стр

Здесь *i* = 4 = *p*, упорядоченный отрезок в файле *S*[0] закончился. Дописываем остаток отрезка из файла *S*[0] в файл *F*[0].

F[0]: 12 14 18 21 29 34 36

F[1]:

i = 4
x[0]
S[0]: 12 14 21 34 [19] 43 48

j = 3
x[1]
S[1]: 18 29 36 [53]

F[0]: 12 14 18 21 29 34 36 53

F[1]:

i = 4
x[0]
S[0]: 12 14 21 34 [19] 43 48

j = 4
x[1]
S[1]: 18 29 36 53 [ ]

стр

Файл *S*[1], а значит и упорядоченный отрезок в файле *S*[1] закончился. Полагаем *n* = 1, *i* = 0, и записываем очередной отрезок из файла *S*[0] в файл *F*[1].

i = 0
x[0]
S[0]: 12 14 21 34 [19] 43 48

j = 4
x[1]
S[1]: 18 29 36 53 [ ]

F[0]:	12	14	18	21	29	34	36	53
F[1]:	19							
						i = 1 *x*[0]		
S[0]:	12	14	21	34	19	43	48	
					j = 4 *x*[1]			
S[1]:	18	29	36	53				

F[0]:	12	14	18	21	29	34	36	53
F[1]:	19	43						
							i = 2 *x*[0]	
S[0]:	12	14	21	34	19	43	48	
					j = 4 *x*[1]			
S[1]:	18	29	36	53				

F[0]:	12	14	18	21	29	34	36	53
F[1]:	19	43	48					
								i = 3 *x*[0]
S[0]:	12	14	21	34	19	43	48	
					j = 4 *x*[1]			
S[1]:	18	29	36	53				

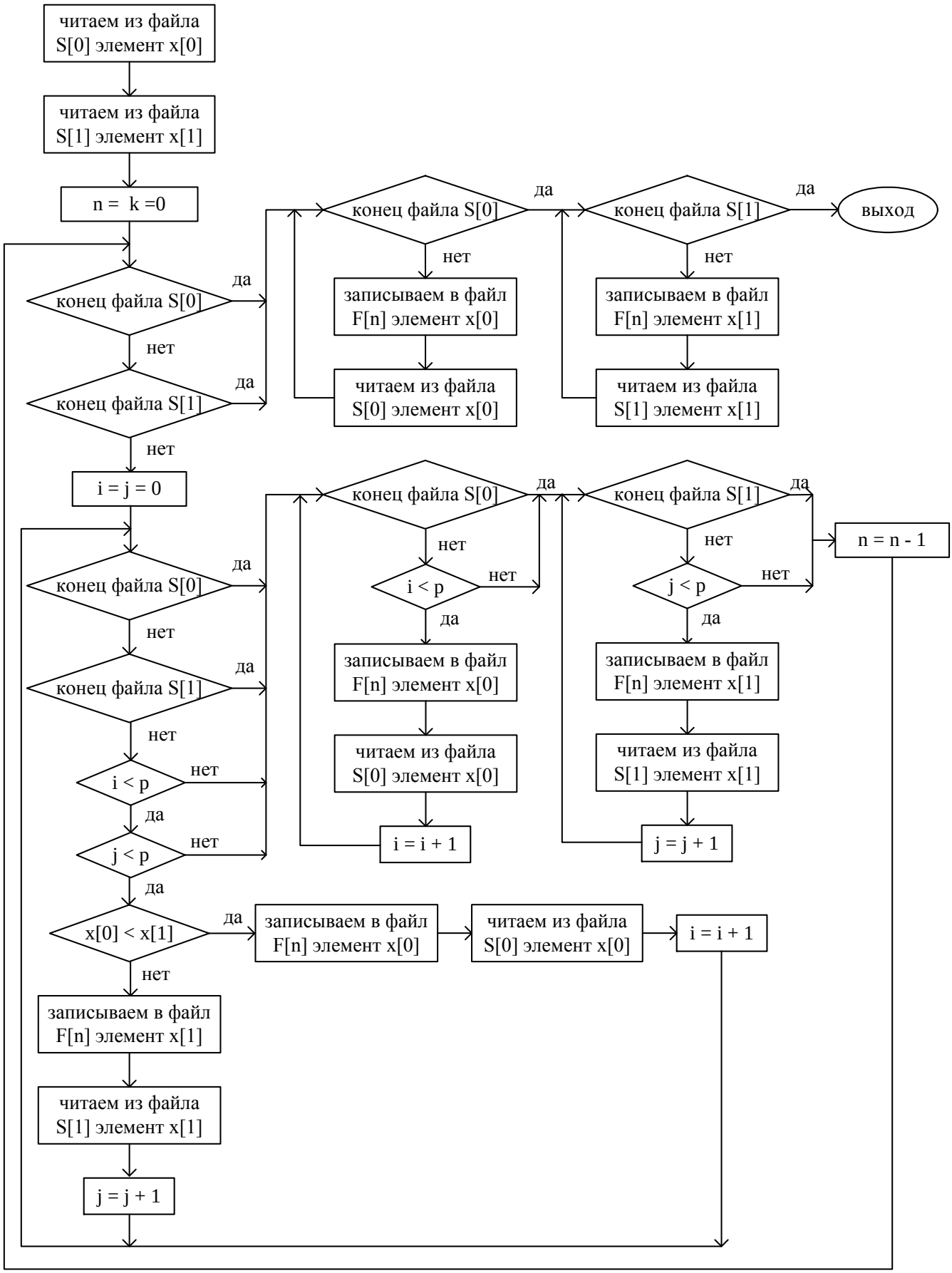
Файл *S*[0] закончился, значит, работа алгоритма завершена.
стр

Алгоритм фазы слияния на четырех файлах

1. Читаем из файла *S*[0] элемент *x*[0].
2. Читаем из файла *S*[1] элемент *x*[1].
3. Полагаем *n* = 0, *k* = 0.
4. Пока не конец файла *S*[0] и не конец файла *S*[1] выполняем:
   4.1. полагаем *i* = *j* = 0;
   4.2. пока не конец файла *S*[0] и не конец файла *S*[1] и пока i < *p* и j < *p* выполняем:
      4.2.1. если *x*[0] < *x*[1], то делаем:
         4.2.1.1 записываем в файл *F*[*n*] элемент *x*[0];
         4.2.1.2. читаем из файла *S*[0] элемент *x*[0];

4.2.1.3. полагаем $i = i + 1$;
4.2.2. иначе:
4.2.2.1. записываем в файл $F[n]$ элемент $x[1]$;
4.2.2.2. читаем из файла $S[1]$элемент $x[1]$
4.2.2.3. полагаем $j = j + 1$;
4.3. пока не конец файла $S[0]$ и $i < p$ выполняем:
4.3.1. записываем в файл $F[n]$ элемент $x[0]$;
4.3.2. читаем из файла $S[0]$ элемент $x[0]$;
4.3.3. полагаем $i = i + 1$;
4.4. пока не конец файла $S[1]$ и $j < p$ выполняем:
4.4.1. записываем в файл $F[n]$ элемент $x[1]$;
4.4.2. читаем из файла $S[1]$ элемент $x[1]$;
4.4.3. полагаем $j = j + 1$;
4.5. полагаем $n = 1 - n$.
5. Пока не конец файла $S[0]$ выполняем:
5.1. записываем в файл $F[n]$ элемент $x[0]$;
5.2. читаем из файла $S[0]$ элемент $x[0]$.
6. Пока не конец файла $S[1]$ выполняем:
6.1. записываем в файл $F[n]$ элемент $x[1]$;
6.2. читаем из файла $S[1]$ элемент $x[1]$.

Блок-схема фазы слияния на четырех файлах



стр

Анализ сортировки с помощью слияния.

Поскольку на каждом проходе p удваивается и сортировка заканчивается при $p \geq n$ (где n – длина файла), то всего требуется $[\log(n)]$ проходов. На каждом проходе по

определению копируются по одному разу все n элементов. Поэтому общее число пересылок M = n [log(n)].

Число сравнений ключей C даже меньше M, поскольку при копировании остатков никаких сравнений не производится. Однако поскольку сортировки слиянием обычно употребляются в ситуациях, где приходится пользоваться внешними запоминающими устройствами, то затраты на операции пересылки на несколько порядков превышают затраты на сравнения. Поэтому детальный анализ числа сравнений особого практического интереса не представляет.

стр

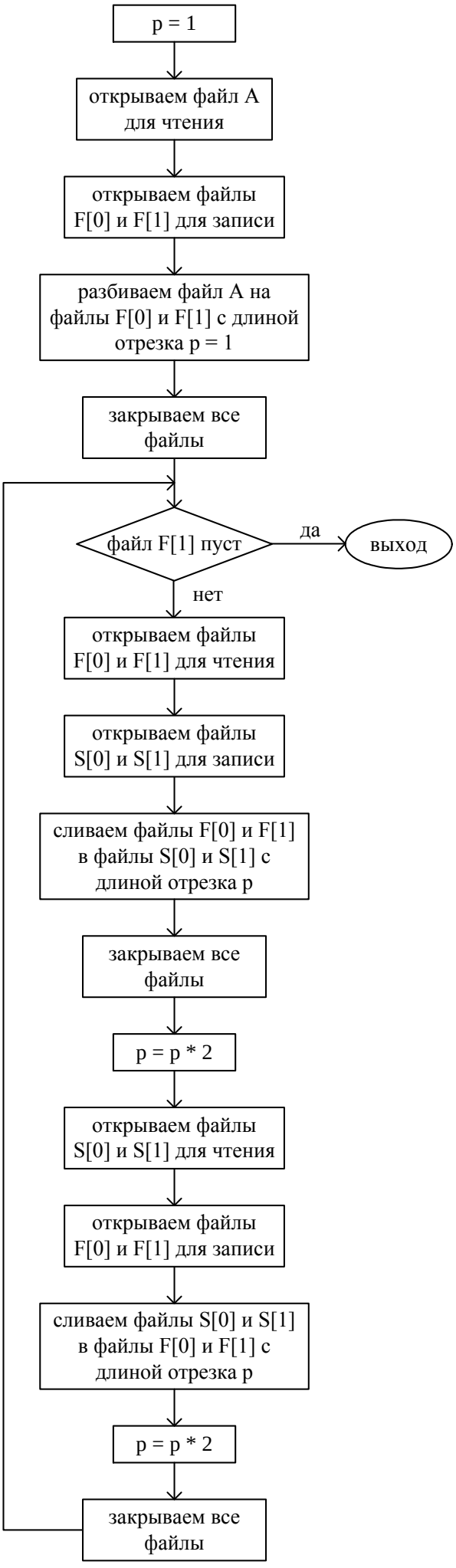
3.2.2.3. Сортировка прямым слиянием на четырех файлах

Непосредственно сортировка состоит из двух этапов: одного повторения фазы разбиения, и затем циклического повторения фазы слияния. Алгоритм можно изложить следующим образом.

Алгоритм прямого слияния на четырех файлах

1. Полагаем p = 1.
2. Открываем файл A для чтения.
3. Открываем файлы F[0] и F[1] для записи.
4. Разбиваем файл A на файлы F[0] и F[1] с длиной отрезка 1.
5. Закрываем все файлы.
6. Если файл F[1] не пуст выполняем:
7. открываем файлы F[0] и F[1] для чтения;
8. открываем файл S[0] и S[1] для записи;
9. сливаем файлы F[0] и F[1] в файлы S[0] и S[1] с длиной отрезка p;
10. закрываем все файлы;
11. полагаем p = 2 p;
12. открываем файлы S[0] и S[1] для чтения;
13. открываем файлы F[0] и F[1] для записи;
14. сливаем файлы S[0] и S[1] в файлы F[0] и F[1] с длиной отрезка p;
15. закрываем все файлы;
16. полагаем p = 2 p;
17. повторяем шаги с 6 по 17.

Блок-схема прямого слияния на четырех файлах


p = 1
открываем файл A для чтения
открываем файлы F[0] и F[1] для записи
разбиваем файл A на файлы F[0] и F[1] с длиной отрезка p = 1
закрываем все файлы
файл F[1] пуст
да
выход
нет
открываем файлы F[0] и F[1] для чтения
открываем файлы S[0] и S[1] для записи
сливаем файлы F[0] и F[1] в файлы S[0] и S[1] с длиной отрезка p
закрываем все файлы
p = p * 2
открываем файлы S[0] и S[1] для чтения
открываем файлы F[0] и F[1] для записи
сливаем файлы S[0] и S[1] в файлы F[0] и F[1] с длиной отрезка p
p = p * 2
закрываем все файлы

стр

3.2.3. Сортировка прямым слиянием на 2N файлах

Идею алгоритма прямого слияния на четырех файлах можно распространить на любое четное (большее двух) количество файлов.

1. Файл *A* разбиваем на *N* файлов: *F*[0], *F*[1],…,*F*[*N*–1]. В файл *F*[0] записываем первый элемент из файла *A*, в файл *F*[1] – второй, и т.д. пока не запишем первые *N* элементов. Затем снова записываем по одному элементу в файл *F*[0], *F*[1] и т.д. пока не закончится файл *A*.

2. Если файл *F*[1] не пуст, сливаем файлы *F*[0], *F*[1],…, *F*[*N*–1] в файлы *S*[0], *S*[1],…, *S*[*N*–1], объединяя первые элементы из *F*[0], *F*[1],…, *F*[*N*–1] в упорядоченный отрезок длины *N* в файле *S*[0], вторые ‘элементы – в упорядоченный отрезок длины *N* в файле *S*[1], и т.д.

3. Если файл *S*[1] не пуст, сливаем файлы *S*[0], *S*[1],…, *S*[*N*–1] в файлы *F*[0], *F*[1],…, *F*[*N*–1], объединяя первые упорядоченные отрезки длины *N* из *S*[0], *S*[1],…, *S*[*N*–1] в упорядоченный отрезок длины N^2 в файле *S*[0], вторые упорядоченные отрезки – в упорядоченный отрезок длины N^2 в файле *S*[1], и т.д.

4. Действуем аналогично шагам 2–3 до тех пор, пока файл *F*[1] или *S*[1] не окажется пустым. При этом будем получать упорядоченные отрезки длины N^3, N^4, и т.д.

Понятно, что при $N > 2$ при каждом проходе мы будем получать упорядоченные отрезки большей длины, чем при прямом слиянии на четырех файлах. Благодаря этому процесс сортировки закончится в общем случае за меньшее число проходов, однако заплатим мы за это большим числом необходимых файлов, а значит, большим объемом используемой памяти.

Приведем пример работы алгоритма при $N = 3$. Рассмотрим файл *A* из предыдущих примеров и разобьем его на файлы *F*[0], *F*[1], *F*[2]. Элементы, которые войдут в разные файлы, выделены разными цветами.

стр

A:	21	12	14	34	18	53	29	36	48	19	43

F[0]:	21	34	29	19
F[1]:	12	18	36	43
F[2]:	14	53	48	

Теперь сливаем полученные последовательности в файлы *S*[0], *S*[1], *S*[2], упорядочивая при этом тройки элементов (расположенные в столбцах и выделенные разными цветами). Первую тройку записываем в файл *S*[0], вторую – в файл *S*[1], третью – в файл *S*[2], четвертую – снова в файл *S*[0], и т.д. до конца файлов *F*[0], *F*[1], *F*[2]. Получаем три последовательности из упорядоченных троек (последняя тройка неполная, т.к. число элементов в исходном файле не кратно трем).

S[0]:	12	14	21	19	43
S[1]:	18	34	53		
S[2]:	29	36	48		

Аналогично сливаем полученные последовательности в файлы *F*[0], *F*[1], *F*[2], объединяя при этом упорядоченные тройки элементов в упорядоченные девятки.

F[0]:	12	14	18	21	29	34	36	48	53
F[1]:	19	43							
F[2]:									

В итоге получено две упорядоченные девятки, причем вторая – неполная. Аналогично сливаем их в файлы *S*[0], *S*[1], *S*[2].

S[0]: 12 14 18 19 21 29 34 36 43 48 53
S[1]:
S[2]:

Файл *S*[1] пуст, работа алгоритма завершена. Упорядоченная последовательность находится в файле *S*[0].

Обратим внимание, что на этот раз сортировка потребовала трех проходов, а не четырех, как в случае трех и четырех файлов.

Стр

3.2.3.1. Фаза разбиения на 2N файлах

На самом деле фаза разбиения сортировки прямым слиянием на 2*N* файлах отличается от фазы разбиения прямым слиянием на четырех файлах (ссылка) лишь правилом изменения индекса *n* – номера файла, в который записывается текущий отрезок. Индекс *n* увеличивается на единицу каждый раз, когда из исходного файла считывается очередной элемент. Когда индекс *n* становится равным *N*, он полагается равным нулю.

Нам потребуются следующие переменные:

x – значение текущего элемента из файла *A*;
F[*N*] – массив из *N* файлов, на которые разбивается файл *A*;
n – номер файла, в который записывается текущий отрезок.

Алгоритм фазы разбиения на 2*N* файлах

1. Читаем из файла *A* элемент x;
2. Полагаем *n* = 0.
3. пока не конец файла *A* выполняем:
   3.1. записываем в файл *F*[*n*] элемент *x*;
   3.2. читаем из файла *A* элемент *x*;
   3.3. полагаем *n* = *n* + 1;
   3.4. если *n* = *N*, полагаем *n* = 0.

Стр

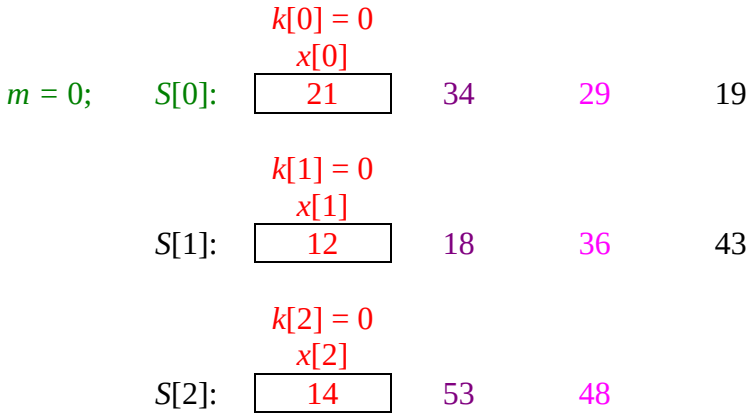
3.2.3.2. Фаза слияния на 2N файлах

Рассмотрим отличие фазы слияния от фазы слияния сортировки прямым слиянием на четырех файлах (ссылка). Они касаются только выбора индексов: *n* – номера файла, в который записывается текущий отрезок, и *m* – номера файла, элемент из которого необходимо записать в файл *F*[*n*]. Индекс *n* меняется так же, как и на фазе разбиения (ссылка). Индекс *m* меняется несколько сложнее: это индекс минимального элемента из тех элементов *x*[0],…, *x*[*N*], где не закончился файл или упорядоченный отрезок длины *p*.

Нам потребуются следующие переменные:

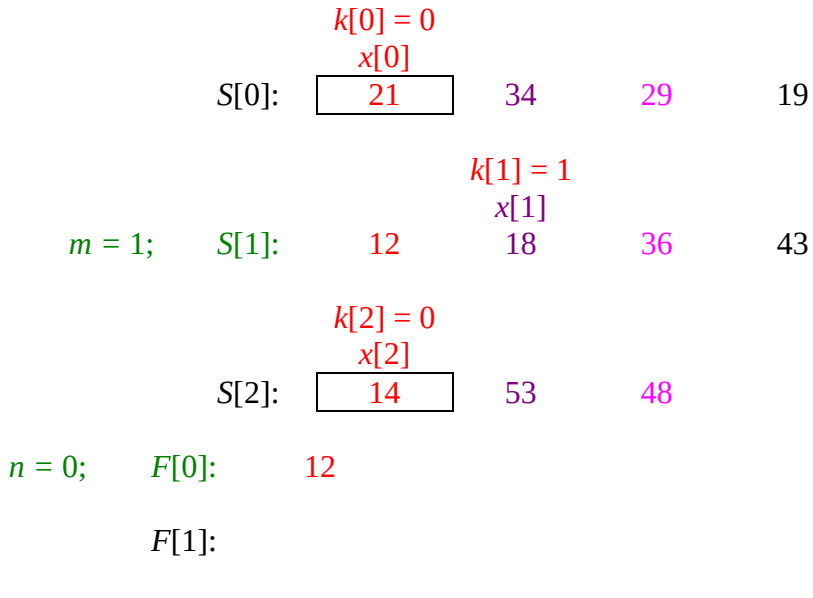
S[*N*] – массив из *N* файлов, из которых производится чтение упорядоченных отрезков;
F[*N*] – массив из *N* файлов, в которые производится запись объединенных упорядоченных отрезков;
x[*N*] – массив из текущих элементов, считанных из файлов *S*[0], *S*[1],…, *S*[*N*];
n – номер файла, в который записывается текущий отрезок;
m – номер файла, элемент из которого записывается в *F*[*n*];
k[*N*] – массив из длин текущих отрезков из файлов *S*[0], *S*[1],…, *S*[*N*];
p – длина отрезка на данном этапе.

Пусть даны три файла $S[0]$, $S[1]$ и $S[2]$, длина упорядоченного отрезка $p = 1$. Сольем их в файлы $F[0]$, $F[1]$ и $F[2]$. Будем считывать элементы из файла $S[i]$ в переменную $x[i]$. Элементы, из которых будем выбирать минимальный элемент, обведены в рамку. Полагаем $n = 0$, т.е. будем записывать отрезок в файл $F[0]$, и $m = 0$. Также полагаем все $k[i] = 0$. Заголовки файлов $F[n]$ и $S[m]$ выделим зеленым цветом.



Стр

Все файлы $S[i]$ не закончились, и все $k[i] < p$, т.е. все упорядоченные отрезки также не закончились. Минимальный элемент среди выделенных – это элемент $x[1]$, поэтому полагаем $m = 1$, записываем элемент $x[m] = 12$ в файл $F[n] = F[0]$, считываем из файла $S[m] = S[1]$ очередной элемент в переменную $x[m]$ и увеличиваем $k[m]$ на единицу. Поскольку теперь $k[1] = 1 = p$, элемент $x[1]$ в выборе минимума не участвует.



Минимальный элемент среди выделенных – это элемент $x[2]$, поэтому полагаем $m = 2$, записываем элемент $x[m] = 14$ в файл $F[n] = F[0]$, считываем из файла $S[m] = S[2]$ очередной элемент в переменную $x[m]$ и увеличиваем $k[m]$ на единицу. Поскольку теперь $k[2] = 1 = p$, элемент $x[2]$ в выборе минимума не участвует.

$k[0] = 0$
$x[0]$

$S[0]$: 21 34 29 19

$k[1] = 1$
$x[1]$

$S[1]$: 12 18 36 43

$k[2] = 1$
$x[2]$

$m = 1$; $S[2]$: 14 53 48

$n = 0$; $F[0]$: 12 14

$F[1]$:

$F[2]$:

Стр

Минимальный элемент среди выделенных – это элемент $x[0]$ (собственно, остался только один выделенный элемент), поэтому полагаем $m = 0$, записываем элемент $x[m] = 21$ в файл $F[n] = F[0]$, считываем из файла $S[m] = S[0]$ очередной элемент в переменную $x[m]$ и увеличиваем $k[m]$ на единицу. Поскольку теперь $k[0] = 1 = p$, элемент $x[0]$ в выборе минимума не участвует.

$k[0] = 1$
$x[0]$

$S[0]$: 21 34 29 19

$k[1] = 1$
$x[1]$

$S[1]$: 12 18 36 43

$k[2] = 1$
$x[2]$

$m = 2$; $S[2]$: 14 53 48

$n = 0$; $F[0]$: 12 14 21

$F[1]$:

$F[2]$:

Все упорядоченные отрезки закончились, значит, полагаем индекс $n = n + 1 = 1$ и начинаем слияние следующих отрезков в файл $F[1]$. Так как все файлы не закончились, полагаем все $k[i] = 0$. Повторяя те же действия, получаем.

$k[0] = 0$
$x[0]$

$S[0]$: 21 34 29 19

$k[1] = 0$
$x[1]$

m = 1; *S*[1]: 12 | 18 | 36 | 43

k[2] = 0
x[2]
S[2]: 14 | 53 | 48

F[0]: 12 14 21

n = 1; *F*[1]: 18

F[2]:

Стр

k[0] = 0
x[0]
m = 0; *S*[0]: 21 | 34 | 29 | 19

k[1] = 1
x[1]
S[1]: 12 18 36 43

k[2] = 0
x[2]
S[2]: 14 | 53 | 48

F[0]: 12 14 21

n = 1; *F*[1]: 18 34

F[2]:

---

k[0] = 1
x[0]
S[0]: 21 34 29 19

k[1] = 1
x[1]
S[1]: 12 18 36 43

k[2] = 0
x[2]
m = 2; *S*[2]: 14 | 53 | 48

F[0]: 12 14 21

n = 1; *F*[1]: 18 34 53

F[2]:

				$k[0] = 1$ $x[0]$	
	$S[0]$:	21	34	29	19
				$k[1] = 1$ $x[1]$	
	$S[1]$:	12	18	36	43
				$k[2] = 1$ $x[2]$	
$m = 2;$	$S[2]$:	14	53	48	

Стр

Все упорядоченные отрезки закончились, значит, полагаем индекс $n = n + 1 = 2$ и начинаем слияние следующих отрезков в файл $F[2]$. Так как все файлы не закончились, полагаем все $k[i] = 0$. Повторяя те же действия, получаем.

				$k[0] = 0$ $x[0]$	
$m = 0;$	$S[0]$:	21	34	29	19
				$k[1] = 0$ $x[1]$	
	$S[1]$:	12	18	36	43
				$k[2] = 0$ $x[2]$	
	$S[2]$:	14	53	48	
	$F[0]$:	12	14	21	
	$F[1]$:	18	34	53	
$n = 2;$	$F[2]$:	29			

					$k[0] = 1$ $x[0]$
	$S[0]$:	21	34	29	19
				$k[1] = 0$ $x[1]$	
$m = 1;$	$S[1]$:	12	18	36	43
				$k[2] = 0$ $x[2]$	
	$S[2]$:	14	53	48	
	$F[0]$:	12	14	21	
	$F[1]$:	18	34	53	
$n = 2;$	$F[2]$:	29	36		

					$k[0] = 1$
					$x[0]$
	$S[0]$:	21	34	29	19
					$k[1] = 1$
					$x[1]$
	$S[1]$:	12	18	36	43
					$k[2] = 1$
					$x[2]$
$m = 2;$	$S[2]$:	14	53	48	
	$F[0]$:	12	14	21	
	$F[1]$:	18	34	53	
$n = 2;$	$F[2]$:	29	36	48	

Стр

Все упорядоченные отрезки закончились, значит, полагаем индекс $n = n + 1 = 3$, т.к. $n = N$, полагаем $n = 0$ и начинаем слияние следующих отрезков в файл $F[0]$. Так как файлы $F[0]$ и $F[1]$ не закончились, а F[2] закончился, полагаем $k[0] = k[1] = 0$, а $k[2] = p$. Повторяя те же действия, получаем.

						$k[0] = 0$
						$x[0]$
$m = 0;$	$S[0]$:	21	34	29		19
						$k[1] = 0$
						$x[1]$
	$S[1]$:	12	18	36		43
						$k[2] = 1$
						$x[2]$
	$S[2]$:	14	53	48		
$n = 0;$	$F[0]$:	12	14	21	19	
	$F[1]$:	18	34	53		
	$F[2]$:	29	36	48		

						$k[0] = 1$
						$x[0]$
	$S[0]$:	21	34	29	19	
						$k[1] = 0$
						$x[1]$
$m = 1;$	$S[1]$:	12	18	36		43

						$k[2] = 1$	
						$x[2]$	
	$S[2]$:	14	53	48			
$n = 0$;	$F[0]$:	12	14	21	19	43	
	$F[1]$:	18	34	53			
	$F[2]$:	29	36	48			

						$k[0] = 1$
						$x[0]$
	$S[0]$:	21	34	29	19	
						$k[1] = 1$
						$x[1]$
	$S[1]$:	12	18	36	43	
						$k[2] = 1$
						$x[2]$
	$S[2]$:	14	53	48		

Все файлы закончились, работа алгоритма завершена.

Стр

Приведем алгоритм фазы слияния. Дополнительно нам потребуется две переменные:

NF – число незакончившихся файлов *S*[0], *S*[1],…, *S*[*N*–1];

NO – число незакончившихся сливаемых упорядоченных отрезков.

Алгоритм фазы слияния

1. Полагаем $i = 0$, $NF = 0$.
2. Пока $i < N$, делаем:
   2.1. читаем элемент $x[i]$ из файла $S[i]$;
   2.2. если не конец файла $S[i]$, делаем:
      2.2.1. полагаем $k[i] = 0$;
      2.2.2. полагаем $NF = NF + 1$;
   2.3. иначе полагаем $k[i] = p$;
   2.4. полагаем $i = i + 1$.
3. Полагаем $n = 0$, $m = 0$.
4. Пока $NF > 0$, делаем:
   4.1. полагаем $NO = NF$;
   4.2. пока $NO > 0$, делаем:
      4.2.1. полагаем $i = 0$, $min = 32765$;
      4.2.2. пока $i < N$, делаем:
         4.2.2.1. если $k[i] < p$ и $x[i] < min$, полагаем $min = x[i]$, $m = i$;
         4.2.2.2. полагаем $i = i + 1$;
      4.2.3. записываем элемент $x[m]$ в файл $F[n]$;
      4.2.4. читаем элемент $x[m]$ из файла $S[m]$;
      4.2.5. полагаем $k[m] = k[m] + 1$;
      4.2.5. если конец файла $S[m]$, делаем:

4.2.5.1. полагаем $k[m] = p$;
4.2.5.2. полагаем $NF = NF - 1$;
4.2.6. если $k[m] = p$, полагаем $NO = NO - 1$;
4.3. полагаем $n = n + 1$;
4.4. если $n = N$, полагаем $n = 0$;
4.5. полагаем $i = 0$;
4.6. пока $i < N$, делаем:
4.6.1. если не конец файла $S[i]$, полагаем $k[i] = 0$;
4.6.2. полагаем $i = i + 1$.
Стр

3.2.3.3. Сортировка прямым слиянием на 2N файлах

Непосредственно сортировка состоит из двух этапов: одного повторения фазы разбиения, и затем циклического повторения фазы слияния. Алгоритм можно изложить следующим образом.

Алгоритм прямого слияния на 2*N* файлах

1. Полагаем $p = 1$.
2. Открываем файл A для чтения.
3. Открываем файлы $F[0], F[1],\ldots, F[N-1]$ для записи.
4. Разбиваем файл A на файлы $F[0], F[1],\ldots, F[N-1]$.
5. Закрываем все файлы.
6. Если файл $F[1]$ не пуст выполняем:
7. открываем файлы $F[0], F[1],\ldots, F[N-1]$ для чтения;
8. открываем файл $S[0], S[1],\ldots, S[N-1]$ для записи;
9. сливаем файлы $F[0], F[1],\ldots, F[N-1]$ в файлы $S[0], S[1],\ldots, S[N-1]$ с длиной отрезка p;
10. закрываем все файлы;
11. полагаем $p = N\,p$;
12. открываем файлы $S[0], S[1],\ldots, S[N-1]$ для чтения;
13. открываем файлы $F[0], F[1],\ldots, F[N-1]$ для записи;
14. сливаем файлы $S[0], S[1],\ldots, S[N-1]$ в файлы $F[0], F[1],\ldots, F[N-1]$ с длиной отрезка p;
15. закрываем все файлы;
16. полагаем $p = N\,p$;
17. повторяем шаги с 6 по 16.

Стр

3.3. Естественное слияние

При использовании стратегии прямого слияния в начале работы мы считаем, что файл состоит из упорядоченных отрезков длины один, т.е. каждый элемент файла представляет собой упорядоченный отрезок. Затем мы объединяем отрезки длины один в отрезки длины два, затем их – в отрезки длины четыре, и т.д., пока файл не будет представлять собой упорядоченный отрезок. При этом мы не учитываем тот факт, что изначально в исходном файле могут присутствовать упорядоченные отрезки длины больше единицы, т.е. мы не получаем никакого преимущества, если данные в файле уже частично упорядочены.

Между тем мы можем работать не с отрезками фиксированной длины, а с упорядоченными отрезками произвольной длины, объединяя два таких отрезка в один так же, как в алгоритмах прямого слияния. При этом можно ожидать, что сортировка

закончится за меньшее число проходов, чем сортировка прямым слиянием. Например, если файл уже упорядочен, это обнаружится за один проход. Основанные на этой идее сортировки называются сортировками естественным слиянием. Как и прямое слияние, такая сортировка может выполняться на трех, четырех и *n* файлах.

стр

3.3.1. Естественное слияние на трех файлах

Идея алгоритма естественного слияния на трех файлах состоит в следующем.

1. Файл *A* разбиваем на два файла: *B* и *C*. В файл *B* записываем первый упорядоченный отрезок из файла *A*, в файл *C* – второй, в файл *B* – третий, в файл *C* – четвертый, и т.д. до конца файла *A*.
2. Если файл *C* не пуст, сливаем файлы *B* и *C* в файл *A*, объединяя по одному упорядоченному отрезку из *B* и *C* в упорядоченный отрезок в файле *A*.
3. Повторяем шаги 1–2, пока это возможно, т.е. пока файл *C* после выполнения первого шага не окажется пустым.

Приведем пример работы алгоритма. Рассмотрим тот же файл *A*, что и в алгоритме прямого слияния на трех файлах. Упорядоченные отрезки, которые войдут в файл *B*, выделены синим цветом, которые войдут в файл *C* – зеленым.

A: 21 12 14 34 18 53 29 36 48 19 43

Разбиваем последовательность *A* на две половины: *B* и *C*.

B: 21 18 53 19 43
C: 12 14 34 29 36 48

Теперь сливаем полученные последовательности в файл *A*, объединяя упорядоченные отрезки (выделенные разными цветами) в упорядоченные отрезки большей длины.

A: 12 14 21 34 18 29 36 48 53 19 43

стр

Опять разбиваем файл *A* на две части: *B* и *C*, которые в *A* выделены разными цветами.

A: 12 14 21 34 18 29 36 48 53 19 43

B: 12 14 21 34 19 43
C: 18 29 36 48 53

Теперь сливаем файлы *B* и *C* в файл *A*, объединяя выделенные разными цветами упорядоченные отрезки.

A: 12 14 18 21 29 34 36 48 53 19 43

Опять разбиваем файл *A* на две части: *B* и *C*, которые в *A* выделены разными цветами.

A: 12 14 18 21 29 34 36 48 53 19 43

B: 12 14 18 21 29 34 36 48 53
C: 19 43

Теперь сливаем файлы *B* и *C* в файл *A*, объединяя выделенные разными цветами упорядоченные отрезки.

A: 12 14 18 19 21 29 34 36 43 48 53

Видно, что теперь файл *A* состоит из одного упорядоченного отрезка. Однако, обнаружить этот факт мы сможем лишь на фазе разбиения: файл *C* окажется пустым.

A: 12 14 18 19 21 29 34 36 43 48 53

B: 12 14 18 19 21 29 34 36 43 48 53
C:

В итоге мы получили упорядоченный файл *A*. Обратим внимание, что на этот раз для сортировки понадобилось три прохода, а не четыре, как в прямом слиянии.

стр

3.3.1.1. Фаза разбиения на трех файлах

Рассмотрим подробнее фазу разбиения файла *A* на файлы *B* и *C*. Напомним идею разбиения: в файл *B* записываем первый упорядоченный отрезок из файла *A*, в файл *C* – второй, в файл *B* – третий, в файл *C* – четвертый, и т.д. до конца файла *A*. В отличие от прямого слияния, здесь нам неизвестны длины упорядоченных отрезков. Определить, закончился отрезок или нет, мы можем, лишь сравнив два соседних элемента файла. Поэтому нам потребуется две переменные:

x_1 – значение текущего элемента из файла *A*;

x_2 – значение следующего элемента из файла *A*.

Если $x_1 > x_2$, то упорядоченный отрезок закончился.

Приведем пример разбиения. Элемент x_1 обведен в рамку, упорядоченные отрезки выделены разными цветами. Сначала считываем первый элемент из файла *A* в переменную x_1.

x_1
A: 12 14 21 34 18 29 36 48 53 19 43

Так как файл *A* не закончился, записываем элемент $x_1 = 12$ в файл *B* и считываем следующий элемент из файла *A* в переменную x_2.

B: 12
C:

x_1 x_2
A: 12 14 21 34 18 29 36 48 53 19 43

стр

Файл *A* не закончился, и при этом $x_1 < x_2$, это означает, что упорядоченный отрезок еще также не закончился. Поэтому полагаем $x_1 = x_2$ и повторяем те же действия: записываем x_1 в файл *B*, читаем элемент из файла *A* в переменную x_2, если не закончился файл или отрезок, то полагаем $x_1 = x_2$.

$x_1 = x_2$
A: 12 14 21 34 18 29 36 48 53 19 43

B: 12 14
C:

		x_1	x_2								
A:	12	14	21	34	18	29	36	48	53	19	43

			$x_1 = x_2$								
A:	12	14	21	34	18	29	36	48	53	19	43

B:	12	14	21
C:			

			x_1	x_2							
A:	12	14	21	34	18	29	36	48	53	19	43

				$x_1 = x_2$							
A:	12	14	21	34	18	29	36	48	53	19	43

B:	12	14	21	34
C:				

				x_1	x_2						
A:	12	14	21	34	18	29	36	48	53	19	43

стр

Файл не закончился, но $x_1 > x_2$, это означает, что упорядоченный отрезок закончился. Поэтому полагаем $x_1 = x_2$, и аналогично записываем второй упорядоченный отрезок в файл *C*.

					$x_1 = x_2$						
A:	12	14	21	34	18	29	36	48	53	19	43

B:	12	14	21	34
C:	18			

					x_1	x_2					
A:	12	14	21	34	18	29	36	48	53	19	43

						$x_1 = x_2$					
A:	12	14	21	34	18	29	36	48	53	19	43

B:	12	14	21	34	
C:	18	29			

стр

						x_1	x_2				
A:	12	14	21	34	18	29	36	48	53	19	43

							$x_1 = x_2$				
A:	12	14	21	34	18	29	36	48	53	19	43

B:	12	14	21	34

C: 18 29 36

							x_1	x_2			
A:	12	14	21	34	18	29	36	48	53	19	43

								$x_1 = x_2$			
A:	12	14	21	34	18	29	36	48	53	19	43

B: 12 14 21 34
C: 18 29 36 48

								x_1	x_2		
A:	12	14	21	34	18	29	36	48	53	19	43

									$x_1 = x_2$		
A:	12	14	21	34	18	29	36	48	53	19	43

B: 12 14 21 34
C: 18 29 36 48 53

									x_1	x_2	
A:	12	14	21	34	18	29	36	48	53	19	43

стр

Файл *A* не закончился, но $x_1 > x_2$, это означает, что второй упорядоченный отрезок закончился. Поэтому полагаем $x_1 = x_2$, и аналогично записываем третий упорядоченный отрезок в файл *B*.

										$x_1 = x_2$	
A:	12	14	21	34	18	29	36	48	53	19	43

B: 12 14 21 34 19
C: 18 29 36 48 53

										x_1	x_2
A:	12	14	21	34	18	29	36	48	53	19	43

											$x_1 = x_2$
A:	12	14	21	34	18	29	36	48	53	19	43

B: 12 14 21 34 19 43
C: 18 29 36 48 53

											x_1	x_2
A:	12	14	21	34	18	29	36	48	53	19	43	

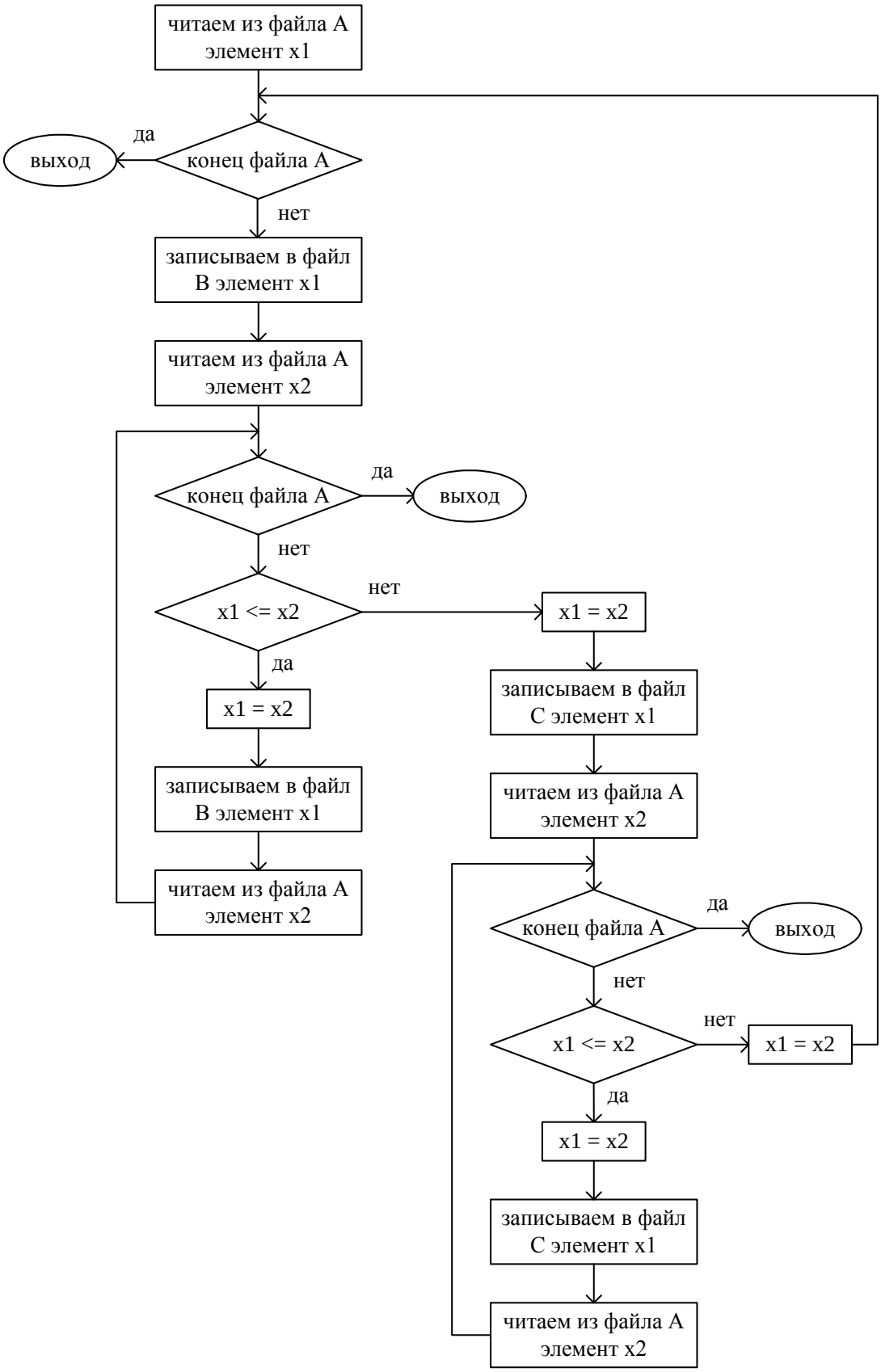
Файл *A* закончился, значит, работа алгоритма завершена.

стр

Алгоритм фазы разбиения на трех файлах

1. Читаем элемент x_1 из файла A.
2. Пока не конец файла A, выполняем шаги 3–8.
   3. Если не конец файла A, делаем:
      4. Записываем элемент x_1 в файл B.
      5. Читаем элемент x_2 из файла A.
      6. Пока не конец файла A и $x_1 \le x_2$, делаем:
         6.1. полагаем $x_1 = x_2$;
         6.2. записываем элемент x_1 в файл B;
         6.3. читаем элемент x_2 из файла A.
   7. Полагаем $x_1 = x_2$.
   8. Если не конец файла A, делаем:
      9. Записываем элемент x_1 в файл C.
      10. Читаем элемент x_2 из файла A.
      11. Пока не конец файла A и $x_1 \le x_2$, делаем:
         11.1. полагаем $x_1 = x_2$;
         11.2. записываем элемент x_1 в файл C;
         11.3. читаем элемент x_2 из файла A.
   12. Полагаем $x_1 = x_2$.

Блок-схема фазы разбиения на трех файлах



стр

3.3.1.2. Фаза слияния на трех файлах

Рассмотрим подробнее фазу слияния файлов *B* и *C* в файл *A*. Здесь, как и на фазе разбиения, нам неизвестны длины упорядоченных отрезков. Определить, закончился

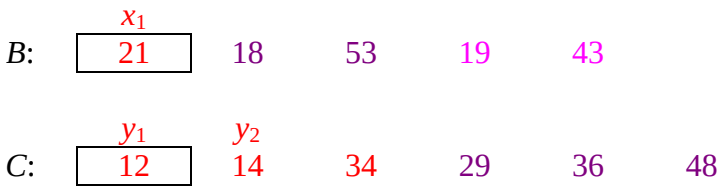
отрезок или нет, мы можем, лишь сравнив два соседних элемента файла. Поэтому для каждого файла нам потребуется две переменные:

x_1 – значение текущего элемента из файла B;
x_2 – значение следующего элемента из файла B;
y_1 – значение текущего элемента из файла C;
y_2 – значение следующего элемента из файла C.

Приведем пример слияния двух файлов. Элементы x_1 и y_1 обведены в рамку, упорядоченные отрезки, подлежащие слиянию, выделены разными цветами. Считываем сначала элементы x_1 и y_1 из файлов B и C соответственно.

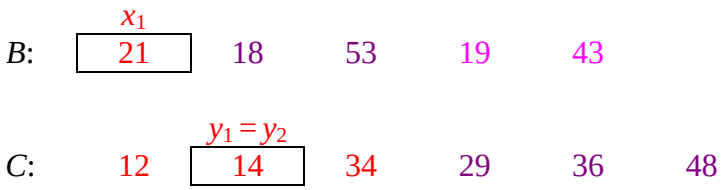
Файлы B и C не пусты, начинаем процесс слияния. Так как $y_1 < x_1$, записываем элемент $y_1 = 12$ в файл A и считываем из файла C следующий элемент в переменную y_2.

A: 12



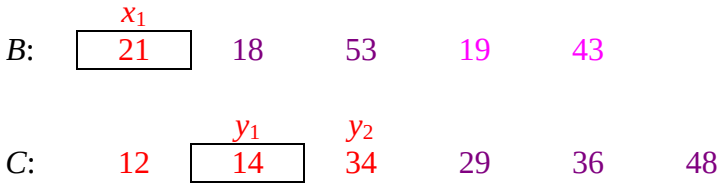
стр

Файл C не закончился, и $y_1 < y_2$, это означает, что упорядоченный отрезок еще также не закончился. Поэтому полагаем $y_1 = y_2$ и повторяем те же действия.

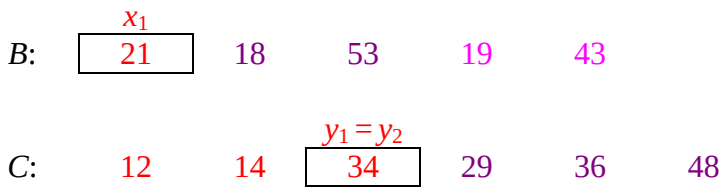


Так как $y_1 < x_1$, записываем элемент $y_1 = 14$ в файл A и считываем из файла C следующий элемент в переменную y_2.

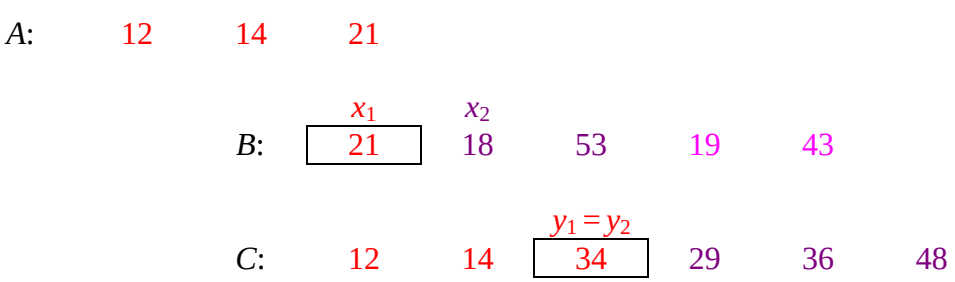
A: 12 14



Файл C не закончился, и $y_1 < y_2$, это означает, что упорядоченный отрезок еще также не закончился. Поэтому полагаем $y_1 = y_2$ и повторяем те же действия.

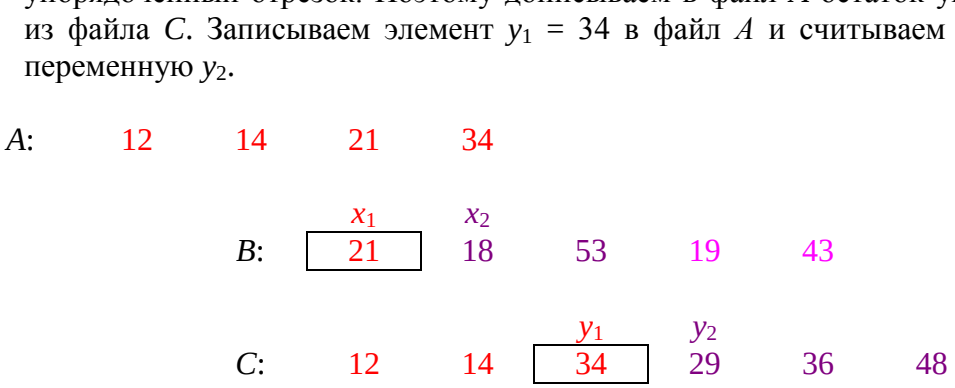


Так как $x_1 < y_1$, записываем элемент $x_1 = 21$ в файл A и считываем из файла B следующий элемент в переменную x_2.

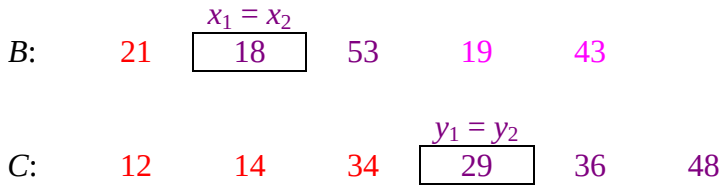


стр

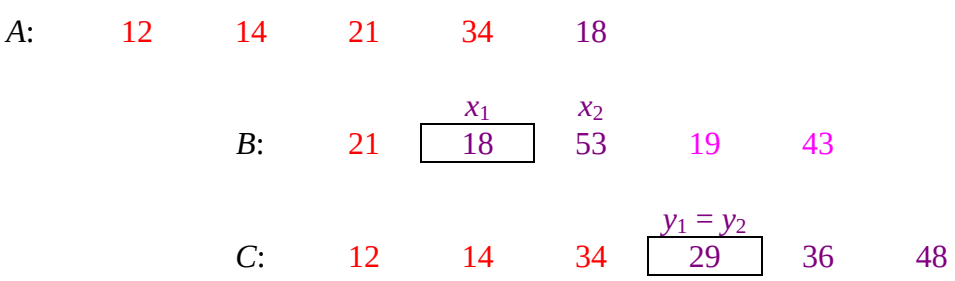
Файл B не закончился, но при этом $x_1 > x_2$, это означает, что закончился упорядоченный отрезок. Поэтому дописываем в файл A остаток упорядоченного отрезка из файла C. Записываем элемент $y_1 = 34$ в файл A и считываем следующий элемент в переменную y_2.



Файл C не закончился, но при этом $y_1 > y_2$, это означает, что закончился упорядоченный отрезок. Полагаем $y_1 = y_2$ и $x_1 = x_2$. Начинаем слияние двух следующих упорядоченных отрезков.



Так как $x_1 < y_1$, записываем элемент $x_1 = 18$ в файл A и считываем из файла B следующий элемент в переменную x_2.



стр

Файл *B* не закончился, и $x_1 < x_2$, это означает, что упорядоченный отрезок еще также не закончился. Поэтому полагаем $x_1 = x_2$ и повторяем те же действия.

			$x_1 = x_2$				
B:	21	18	53	19	43		
				$y_1 = y_2$			
C:	12	14	34	29	36	48	

Так как $y_1 < x_1$, записываем элемент $y_1 = 29$ в файл *A* и считываем из файла *C* следующий элемент в переменную y_2.

A:	12	14	21	34	18	29	
			$x_1 = x_2$				
B:	21	18	53	19	43		
				y_1	y_2		
C:	12	14	34	29	36	48	

Файл *C* не закончился, и $y_1 < y_2$, это означает, что упорядоченный отрезок еще также не закончился. Поэтому полагаем $y_1 = y_2$ и повторяем те же действия.

			$x_1 = x_2$				
B:	21	18	53	19	43		
					$y_1 = y_2$		
C:	12	14	34	29	36	48	

Так как $y_1 < x_1$, записываем элемент $y_1 = 36$ в файл *A* и считываем из файла *C* следующий элемент в переменную y_2.

A:	12	14	21	34	18	29	36
			$x_1 = x_2$				
B:	21	18	53	19	43		
					y_1	y_2	
C:	12	14	34	29	36	48	

стр

Файл *C* не закончился, и $y_1 < y_2$, это означает, что упорядоченный отрезок еще также не закончился. Поэтому полагаем $y_1 = y_2$ и повторяем те же действия.

			$x_1 = x_2$				
B:	21	18	53	19	43		
						$y_1 = y_2$	
C:	12	14	34	29	36	48	

Так как $y_1 < x_1$, записываем элемент $y_1 = 48$ в файл *A* и считываем из файла *C* следующий элемент в переменную y_2.

A: 12 14 21 34 18 29 36 48

$x_1 = x_2$

B: 21 18 [53] 19 43

y_1 y_2

C: 12 14 34 29 36 [48]

Файл *C* закончился, поэтому далее переписываем остаток файла *B* в файл *A*. Записываем элемент x_1 в файл *A*, затем считываем очередной элемент из файла *B* в переменную x_1. Повторяем эти действия до тех пор, пока не дойдем до конца файла *B*.

x_1

B: 21 18 [53] 19 43

A: 12 14 21 34 18 29 36 48 53

---

x_1

B: 21 18 53 [19] 43

A: 12 14 21 34 18 29 36 48 53

---

x_1

B: 21 18 53 19 [43]

A: 12 14 21 34 18 29 36 48 53 19 43

---

x_1

B: 21 18 53 19 43 [ ]

Достигнут конец файла *B*, слияние закончено.

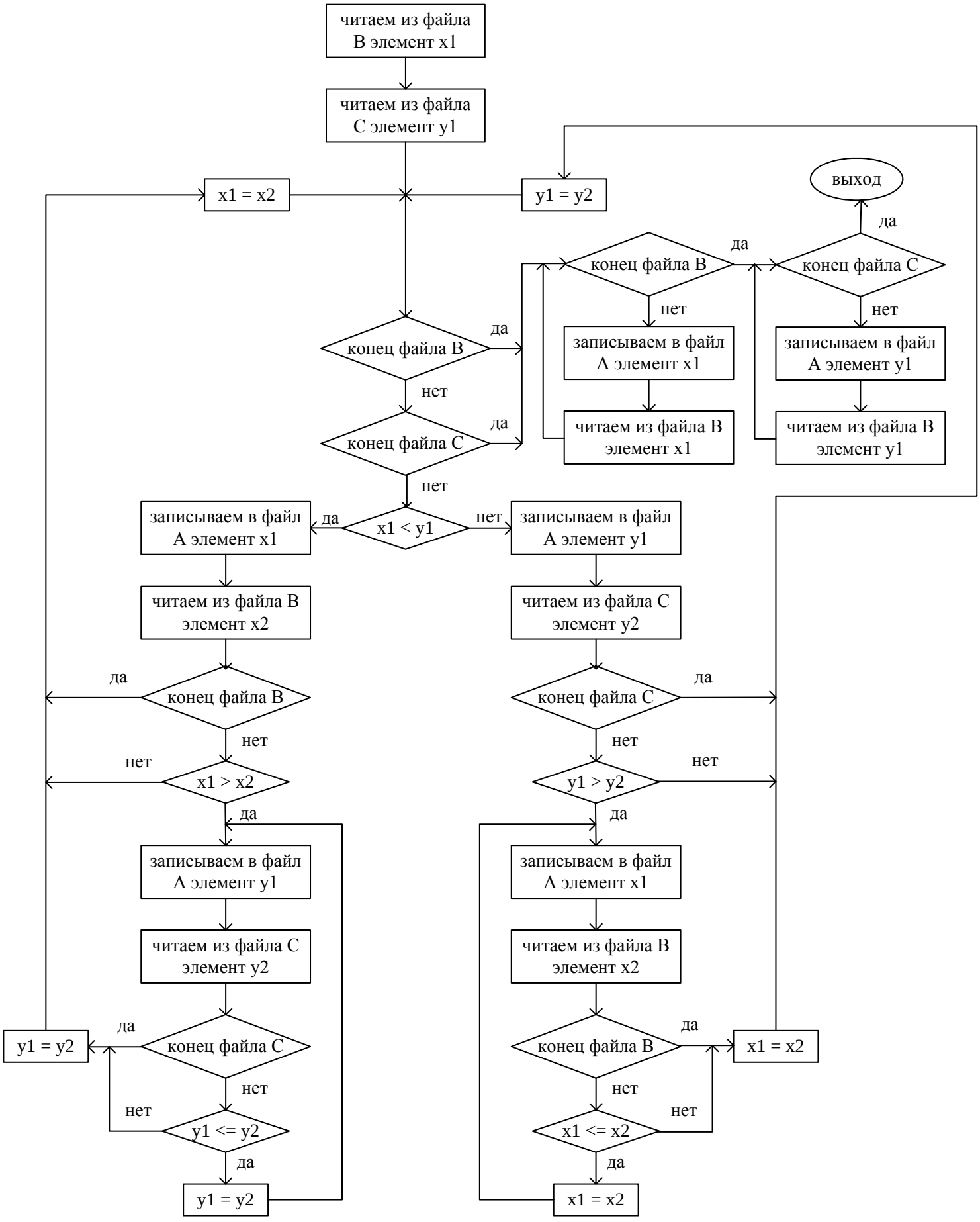
стр

Алгоритм фазы слияния на трех файлах

1. Читаем элемент x_1 из файла *B* и элемент y_1 из файла *C*.
2. Пока не конец файла *B* и не конец файла *C*, выполняем шаги 3–4, затем выполняем шаги 5–6.
3. Если $x_1 < y_1$, делаем:
   - 3.1. записываем элемент x_1 в файл *A*;
   - 3.2. читаем элемент x_2 из файла *B*;
   - 3.3. если не конец файла *B* и $x_1 > x_2$, делаем:
     - 3.3.1. записываем элемент y_1 в файл *A*;
     - 3.3.2. читаем элемент y_2 из файла *C*;
     - 3.3.3. пока не конец файла *C* и $y_1 \le y_2$, делаем:
       - 3.3.3.1. полагаем $y_1 = y_2$;
       - 3.3.3.2. записываем элемент y_1 в файл *A*;
       - 3.3.3.3. читаем элемент y_2 из файла *C*;
     - 3.3.4. полагаем $y_1 = y_2$;
   - 3.4. полагаем $x_1 = x_2$;

4. иначе делаем:
    4.1. записываем элемент y_1 в файл *A*;
    4.2. читаем элемент y_2 из файла *C*;
    4.3. если не конец файла *C* и $y_1 > y_2$, делаем:
        4.3.1. записываем элемент x_1 в файл *A*;
        4.3.2. читаем элемент x_2 из файла *B*;
        4.3.3. пока не конец файла *B* и $x_1 \leq x_2$, делаем:
            4.3.3.1. полагаем $x_1 = x_2$;
            4.3.3.2. записываем элемент x_1 в файл *A*;
            4.3.3.3. читаем элемент x_2 из файла *B*;
        4.3.4. полагаем $x_1 = x_2$;
    4.4. полагаем $y_1 = y_2$.
5. Пока не конец файла *B*, делаем:
    5.1. записываем элемент x_1 в файл *A*;
    5.2. читаем элемент x_1 из файла *B*.
6. Пока не конец файла *C*, делаем:
    6.1. записываем элемент y_1 в файл *A*;
    6.2. читаем элемент y_1 из файла *C*.

Блок-схема фазы слияния на трех файлах


читаем из файла
B элемент x1
читаем из файла
C элемент y1
x1 = x2
y1 = y2
выход
да
конец файла B
да
конец файла C
нет
нет
конец файла B
да
записываем в файл
A элемент x1
записываем в файл
A элемент y1
нет
конец файла C
да
читаем из файла B
элемент x1
читаем из файла B
элемент y1
нет
записываем в файл
A элемент x1
да
x1 < y1
нет
записываем в файл
A элемент y1
читаем из файла B
элемент x2
читаем из файла C
элемент y2
да
конец файла B
да
конец файла C
нет
нет
x1 > x2
нет
y1 > y2
нет
да
да
записываем в файл
A элемент y1
записываем в файл
A элемент x1
читаем из файла C
элемент y2
читаем из файла B
элемент x2
да
y1 = y2
да
конец файла C
да
конец файла B
x1 = x2
нет
нет
y1 <= y2
нет
x1 <= x2
нет
да
да
y1 = y2
x1 = x2

стр

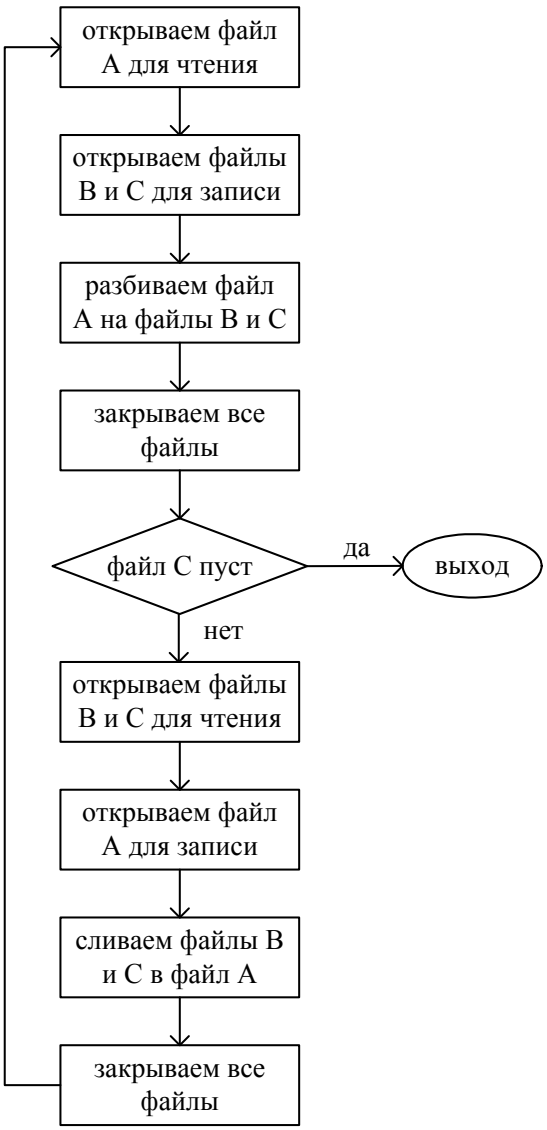
3.3.1.3. Сортировка естественным слиянием на трех файлах

Как уже было сказано, сама сортировка состоит из повторения фазы разбиения и фазы слияния. Алгоритм можно изложить следующим образом.

Алгоритм сортировки естественным слиянием на трех файлах

1. Открываем файл *A* для чтения.
2. Открываем файлы *B* и *C* для записи.
3. Разбиваем файл *A* на файлы *B* и *C*.
4. Закрываем все файлы.
5. Если файл *C* пуст, работа алгоритма закончена.
6. Открываем файлы *B* и *C* для чтения.
7. Открываем файл *A* для записи.
8. Сливаем файлы *B* и *C* в файл *A*.
9. Закрываем все файлы.
10. Повторяем алгоритм с шага 1.

Блок-схема сортировки естественным слиянием на трех файлах



стр

3.3.2. Естественное слияние на четырех файлах

Идея алгоритма естественного слияния на четырех файлах состоит в следующем. В отличие от слияния на трех файлах, фаза разбиения выполняется лишь один раз. Затем слияние файлов совмещается с разбиением: получившиеся после слияния упорядоченные отрезки записываются не в один исходный файл, а в два файла, первый, третий и т.д. отрезки – в один файл, второй, четвертый и т.д. отрезки – во второй файл.

Более формально идею можно изложить следующим образом.

1. Файл *A* разбиваем на два файла: *C* и *D*. В файл *C* записываем первый упорядоченный отрезок из файла *A*, в файл *D* – второй, в файл *C* – третий, в файл *D* – четвертый, и т.д. до конца файла *A*.
2. Если файл *D* не пуст, сливаем файлы *C* и *D* в файлы *A* и *B*, объединяя первые упорядоченные отрезки из *C* и *D* в упорядоченный отрезок в файле *A*, вторые упорядоченные отрезки из *C* и *D* в упорядоченный отрезок в файле *B*, и т.д.
3. Если файл *B* не пуст, сливаем файлы *A* и *B* в файлы *C* и *D*, объединяя первые упорядоченные отрезки из *A* и *B* в упорядоченный отрезок в файле *C*, вторые упорядоченные отрезки из *A* и *B* в упорядоченный отрезок в файле *D*, и т.д.
4. Повторяем шаги 2–3, пока это возможно, т.е. пока файл *B* или *D* не окажется пустым.

Приведем пример работы алгоритма. Рассмотрим файл *A* из предыдущих примеров. Упорядоченные отрезки, которые войдут в файл *C*, выделены синим цветом, которые войдут в файл *D* – зеленым.

A:	21	12	14	34	18	53	29	36	48	19	43

Разбиваем файл *A* на две части: *B* и *C*.

C:	21	18	53	19	43	
D:	12	14	34	29	36	48

Теперь сливаем полученные последовательности в файлы *A* и *B*, объединяя упорядоченные отрезки (выделенные разными цветами) в упорядоченные отрезки большей длины. Обратим внимание, что у последнего упорядоченного отрезка в файле *C* нет парного отрезка в файле *D*, поэтому он целиком переписывается в очередной файл *A*.

A:	12	14	21	34	19	43
B:	18	29	36	48	53	

стр

Аналогично сливаем файлы *A* и *B* в файлы *C* и *D*.

A:	12	14	21	34	19	43
B:	18	29	36	48	53	

C:	12	14	18	21	29	34	36	48	53
D:	19	43							

Аналогично сливаем файлы *C* и *D* в файлы *A* и *B*.

C:	12	14	18	21	29	34	36	48	53
D:	19	43							

A: 12 14 18 19 21 29 34 36 43 48 53
B:

Файл *B* оказался пустым, значит, работа алгоритма закончена. Упорядоченная последовательность содержится в файле *A*.

Сортировка выполнены за три прохода, как и в случае естественного слияния на трех файлах. Однако здесь мы получаем выигрыш по скорости, т.к. фаза разбиения как таковая выполняется только один раз, в последующих проходах она объединена с фазой слияния.
стр

3.3.2.1. Фаза разбиения на четырех файлах

На самом деле фаза разбиения сортировки естественным слиянием на четырех файлах ничем не отличается от фазы разбиения естественным слиянием на трех файлах (ссылка). Но, как вы могли заметить, в блок-схеме и в словесном описании алгоритма повторяются одинаковые части, отличающиеся лишь именем файла. Чтобы избежать повторов и сделать описание более компактным, мы создадим массив, состоящий из файлов, куда будет вестись запись упорядоченных отрезков.

Нам потребуется следующие переменные:
x_1 – значение текущего элемента из файла *A*;
x_2 – значение следующего элемента из файла *A*;
F[2] – массив из двух файлов, на которые разбивается файл *A*;
n – номер файла, в который записывается текущий отрезок.

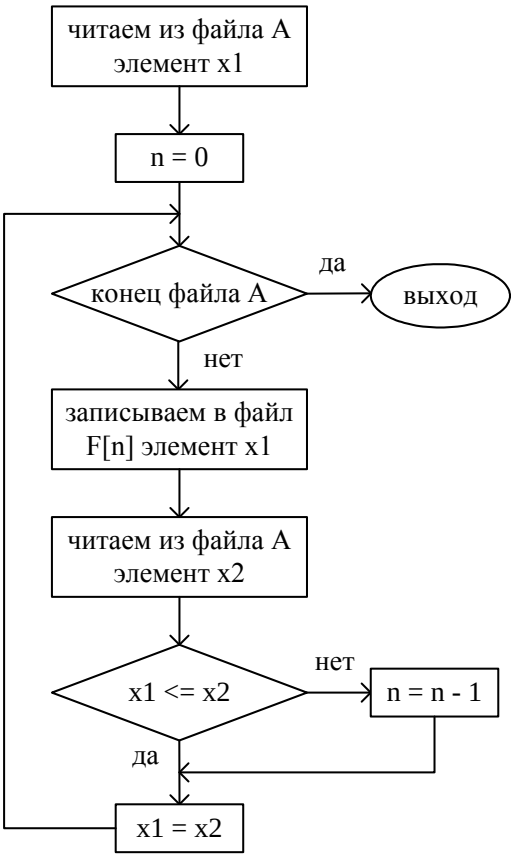
Индекс *n* меняется с 0 на 1 и с 1 на 0, т.е. для его изменения нужно положить $n = 1 - n$. Изменение индекса происходит каждый раз, когда заканчивается очередной упорядоченный отрезок, т.е. когда $x_1 > x_2$.

Алгоритм фазы разбиения на четырех файлах

1. Читаем элемент x_1 из файла *A*.
2. Полагаем $n = 0$.
3. Пока не конец файла *A*, делаем:
   3.1. записываем элемент x_1 в файл *F*[*n*];
   3.2. читаем элемент x_2 из файла *A*;
   3.3. если $x_1 > x_2$, полагаем $n = 1 - n$;
   3.4. полагаем $x_1 = x_2$.

Конечно, этот вариант алгоритма можно использовать и в прямом слиянии на трех файлах.

Блок-схема фазы разбиения



стр

3.3.2.2. Фаза слияния на четырех файлах

Рассмотрим подробнее фазу слияния. Как и на фазе разбиения, будем использовать массивы файлов. Нам потребуются следующие переменные:

S[2] – массив из двух файлов, из которых производится чтение упорядоченных отрезков;

F[2] – массив из двух файлов, в которые производится запись объединенных упорядоченных отрезков;

x[2] – массив из текущих элементов, считанных из файлов *S*[0], *S*[1];

y[2] – массив из следующих элементов файлов *S*[0], *S*[1];

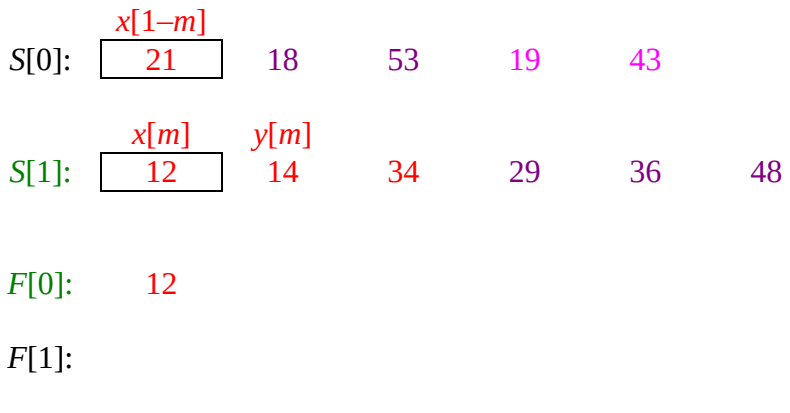
n – номер файла, в который записывается текущий отрезок;

m – номер файла, элемент из которого необходимо записать в файл *F*[*n*].

Приведем пример слияния двух файлов. Элементы *x*[0] и *x*[1] обведены в рамку, упорядоченные отрезки, подлежащие слиянию, выделены разными цветами. Считываем сначала элементы *x*[0] и *x*[1] из файлов *S*[0] и *S*[1] соответственно. Полагаем *n* = 0, *m* = 0. Это означает, что сливать отрезки мы будем в файл *F*[0], а записывать очередной элемент будем из файла *S*[0]. Но если окажется, что наименьший элемент находится не в файле *S*[0], нужно будет изменить индекс *m*, т.е. положить *m* = 1 – *m*. Будем выделять заголовки файлов *S*[*m*] и *F*[*n*] зеленым цветом.

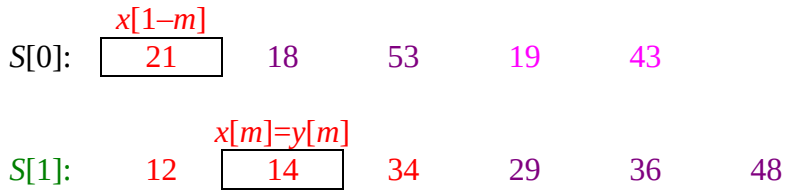
	x[*m*]					
S[0]:	21	18	53	19	43	
	x[1–*m*]					
S[1]:	12	14	34	29	36	48

Файлы $S[0]$ и $S[1]$ не пусты, начинаем процесс слияния. Так как $x[m] > x[1–m]$, полагаем индекс $m = 1 – m$, записываем элемент $x[m] = 12$ в файл $F[n] = F[0]$ и считываем из файла $S[m] = S[1]$ следующий элемент в переменную $y[m]$.

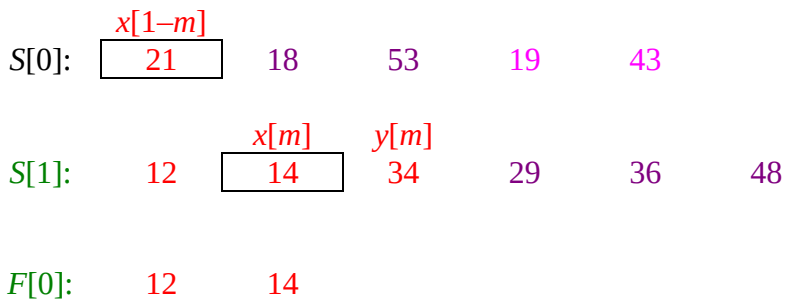


стр

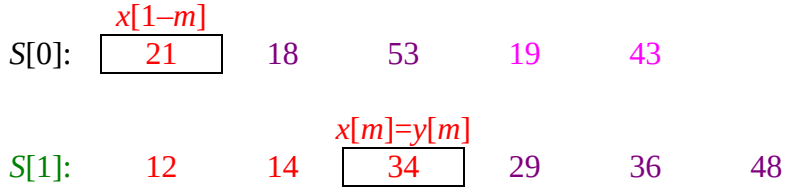
Файл $S[m] = S[1]$ не закончился, и при этом $x[m] < y[m]$, значит, упорядоченный отрезок также не закончился. Полагаем $x[m] = y[m]$ и повторяем те же действия.



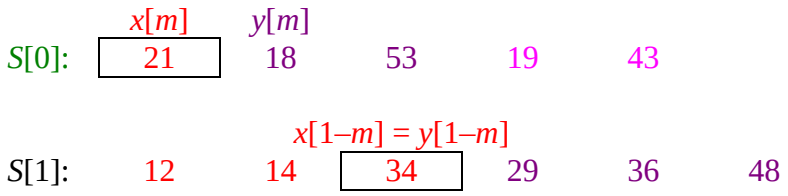
Так как $x[m] < x[1–m]$, записываем элемент $x[m] = 14$ в файл $F[0]$ и считываем из файла $S[m]$ следующий элемент в переменную $y[m]$.



Файл $S[m] = S[1]$ не закончился, и при этом $x[m] < y[m]$, значит, упорядоченный отрезок также не закончился. Полагаем $x[m] = y[m]$ и повторяем те же действия.

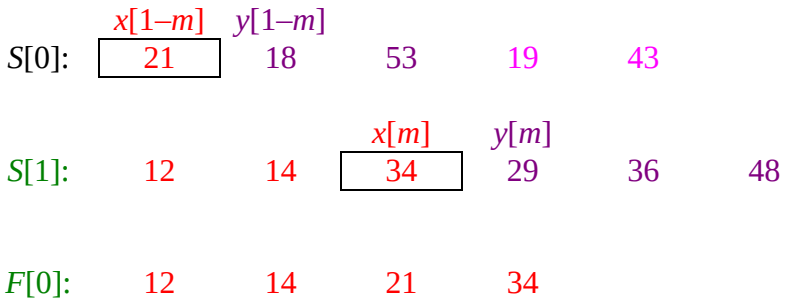


Так как $x[m] > x[1–m]$, полагаем индекс $m = 1 – m$, записываем элемент $x[m] = 21$ в файл $F[0]$ и считываем из файла $S[m]$ следующий элемент в переменную $y[m]$.

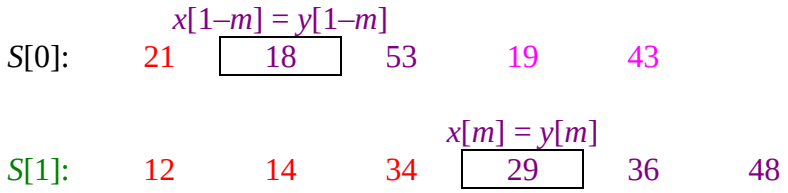


стр

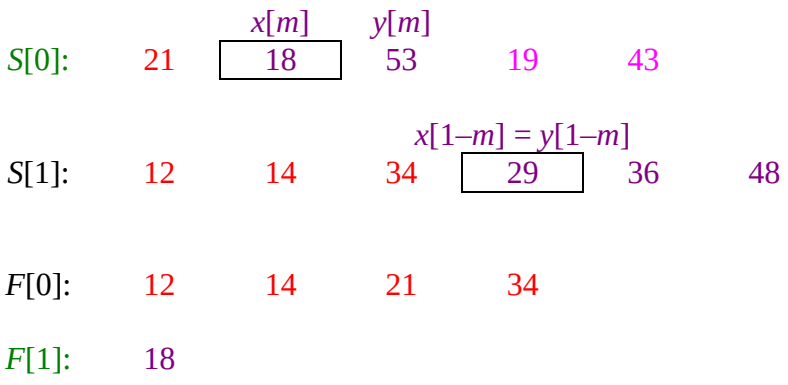
Файл $S[m] = S[0]$ не закончился, но при этом $x[m] > y[m]$, значит, один упорядоченный отрезок закончился. Полагаем индекс $m = 1 - m$, и дописываем остаток упорядоченного отрезка из файла $S[m] = S[1]$ в файл $F[0]$. При считывании очередного элемента каждый раз проверяем, не закончился ли файл $S[m]$. Записываем элемент $x[m] = 34$ в файл $F[0]$ и считываем из файла $S[m]$ следующий элемент в переменную $y[m]$.



Файл $S[m] = S[1]$ не закончился, но при этом $x[m] > y[m]$, значит, второй упорядоченный отрезок закончился. Полагаем $x[m] = y[m]$ и $x[1–m] = y[1–m]$, и начинаем слияние двух следующих упорядоченных отрезков в файл $F[1]$, для чего полагаем $n = 1 - n$.



Так как $x[m] > x[1–m]$, полагаем индекс $m = 1 - m$, записываем элемент $x[m] = 18$ в файл $F[n] = F[1]$ и считываем из файла $S[m] = S[0]$ следующий элемент в переменную $y[m]$.



стр

Файл $S[m] = S[0]$ не закончился, и при этом $x[m] < y[m]$, значит, упорядоченный отрезок также не закончился. Полагаем $x[m] = y[m]$ и повторяем те же действия.

			$x[m] = y[m]$		
$S[0]$:	21	18	53	19	43

				$x[1–m] = y[1–m]$		
$S[1]$:	12	14	34	29	36	48

Так как $x[m] > x[1–m]$, полагаем индекс $m = 1 – m$, записываем элемент $x[m] = 29$ в файл $F[n] = F[1]$ и считываем из файла $S[m] = S[1]$ следующий элемент в переменную $y[m]$.

			$x[1–m] = y[1–m]$		
$S[0]$:	21	18	53	19	43

				$x[m]$	$y[m]$	
$S[1]$:	12	14	34	29	36	48

$F[0]$:	12	14	21	34
$F[1]$:	18	29		

Файл $S[m] = S[1]$ не закончился, и при этом $x[m] < y[m]$, значит, упорядоченный отрезок также не закончился. Полагаем $x[m] = y[m]$ и повторяем те же действия.

			$x[1–m] = y[1–m]$		
$S[0]$:	21	18	53	19	43

					$x[m] = y[m]$	
$S[1]$:	12	14	34	29	36	48

Так как $x[m] < x[1–m]$, записываем элемент $x[m] = 36$ в файл $F[n] = F[1]$ и считываем из файла $S[m] = S[1]$ следующий элемент в переменную $y[m]$.

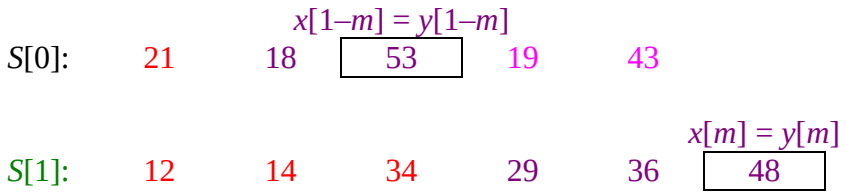
			$x[1–m] = y[1–m]$		
$S[0]$:	21	18	53	19	43

					$x[m]$	$y[m]$
$S[1]$:	12	14	34	29	36	48

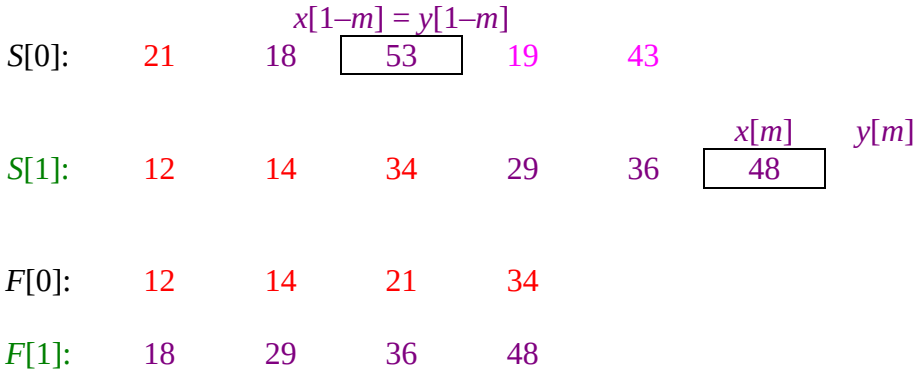
$F[0]$:	12	14	21	34
$F[1]$:	18	29	36	

стр

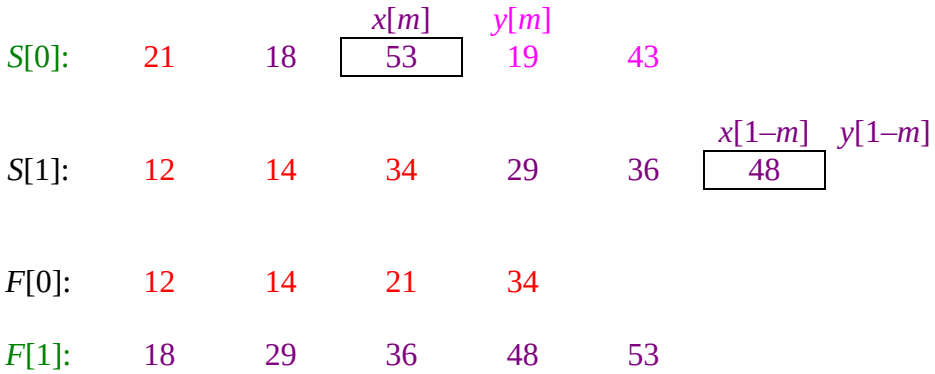
Файл $S[m] = S[1]$ не закончился, и при этом $x[m] < y[m]$, значит, упорядоченный отрезок также не закончился. Полагаем $x[m] = y[m]$ и повторяем те же действия.



Так как $x[m] < x[1–m]$, записываем элемент $x[m] = 48$ в файл $F[n] = F[1]$ и считываем из файла $S[m] = S[1]$ следующий элемент в переменную $y[m]$.



Файл $S[m] = S[1]$ закончился, значит, закончился и первый упорядоченный отрезок. Полагаем индекс $m = 1 – m$ и дописываем в файл $F[1]$ остаток упорядоченного отрезка из файла $S[m] = S[0]$. Записываем элемент $x[m] = 53$ в файл $F[n] = F[1]$ и считываем из файла $S[m]$ следующий элемент в переменную $y[m]$.



стр

Файл $S[m] = S[0]$ не закончился, но при этом $x[m] > y[m]$, значит, второй упорядоченный отрезок закончился. Полагаем индекс $n = 1 – n$, элемент $x[m] = y[m]$, и далее считываем элементы только из файла $S[m] = S[0]$. Записываем первый упорядоченный отрезок в файл $F[n] = F[0]$, затем, если файл $S[m]$ не закончился, полагаем индекс $n = 1 – n$, и записываем второй упорядоченный отрезок в файл $F[n] = F[1]$, и т.д. до конца файла $S[m]$.

Записываем элемент $x[m] = 19$ в файл $F[n] = F[0]$ и считываем из файла $S[m] = S[0]$ следующий элемент в переменную $y[m]$.

				x[*m*]	*y*[*m*]
S[0]:	21	18	53	19	43

F[0]:	12	14	21	34	19
F[1]:	18	29	36	48	53

Файл *S*[*m*] = *S*[0] не закончился, и при этом *x*[*m*] < *y*[*m*], значит, упорядоченный отрезок также не закончился. Полагаем *x*[*m*] = *y*[*m*] и повторяем те же действия.

					x[*m*] = *y*[*m*]
S[0]:	21	18	53	19	43

Записываем элемент *x*[*m*] = 43 в файл *F*[*n*] = *F*[0] и считываем из файла *S*[*m*] = *S*[0] следующий элемент в переменную *y*[*m*].

				x[*m*]	*y*[*m*]
S[0]:	21	18	53	19	43

F[0]:	12	14	21	34	19	43
F[1]:	18	29	36	48	53	

Файл *S*[*m*] закончился, работа алгоритма завершена.

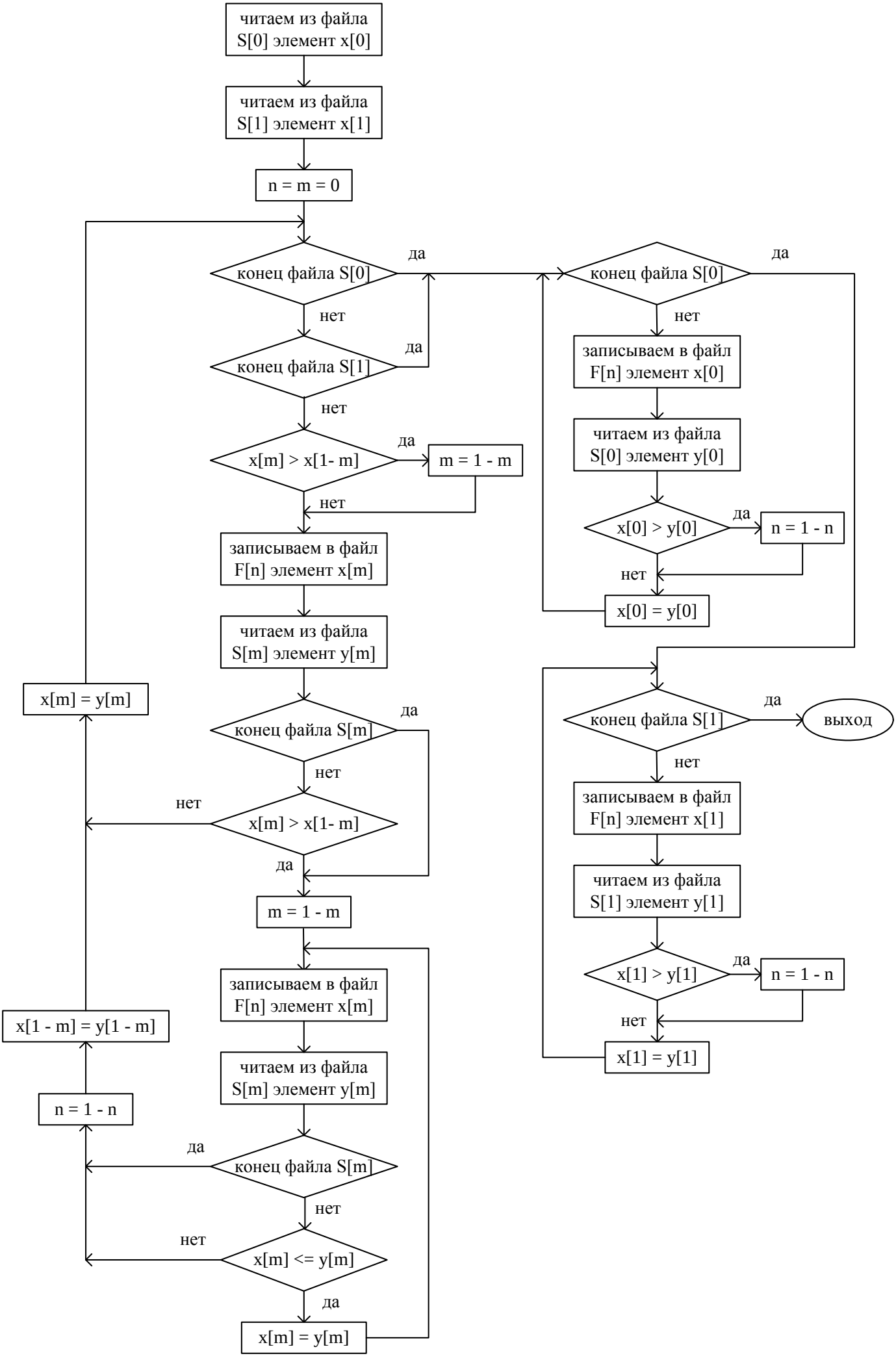
стр

Алгоритм фазы слияния на четырех файлах

1. Читаем элемент *x*[0] из файла *S*[0].
2. Читаем элемент *x*[1] из файла *S*[1].
3. Полагаем *n* = 0, *m* = 0.
4. Пока не конец файла *S*[0] и не конец файла *S*[1], делаем:
   4.1. если *x*[*m*] > *x*[1–*m*], полагаем *m* = 1 – *m*;
   4.2. записываем элемент *x*[*m*] в файл *F*[*n*];
   4.3. читаем элемент *y*[*m*] из файла *S*[*m*];
   4.4. если конец файла *S*[*m*] или *x*[*m*] > *y*[*m*], делаем:
      4.4.1. полагаем *m* = 1 – *m*;
      4.4.2. записываем элемент *x*[*m*] в файл *F*[*n*];
      4.4.3. читаем элемент *y*[*m*] из файла *S*[*m*];
      4.4.4. пока не конец файла *S*[*m*] и *x*[*m*] ≤ *y*[*m*], делаем:
         4.4.4.1. полагаем *x*[*m*] = *y*[*m*];
         4.4.4.2. записываем элемент *x*[*m*] в файл *F*[*n*];
         4.4.4.3. читаем элемент *y*[*m*] из файла *S*[*m*];
      4.4.5. полагаем *x*[1–*m*] = *y*[1–*m*];
      4.4.6. полагаем *n* = 1 – *n*;
   4.5. полагаем *x*[*m*] = *y*[*m*].
5. Пока не конец файла *S*[0], делаем:
   5.1. записываем элемент *x*[0] в файл *F*[*n*];
   5.2. читаем элемент *y*[0] из файла *S*[0];
   5.3. если *x*[0] > *y*[0], полагаем *n* = 1 – *n*;
   5.4. полагаем *x*[0] = *y*[0].

6. Пока не конец файла $S[1]$, делаем:
   6.1. записываем элемент $x[1]$ в файл $F[n]$;
   6.2. читаем элемент $y[1]$ из файла $S[1]$;
   6.3. если $x[1] > y[1]$, полагаем $n = 1 - n$;
   6.4. полагаем $x[1] = y[1]$.

Блок-схема фазы слияния на четырех файлах


читаем из файла S[0] элемент x[0]
читаем из файла S[1] элемент x[1]
n = m = 0
конец файла S[0]
да
нет
конец файла S[1]
да
нет
x[m] > x[1- m]
да
m = 1 - m
нет
записываем в файл F[n] элемент x[m]
читаем из файла S[m] элемент y[m]
конец файла S[m]
да
нет
x[m] > x[1- m]
нет
да
m = 1 - m
записываем в файл F[n] элемент x[m]
читаем из файла S[m] элемент y[m]
конец файла S[m]
да
нет
x[m] <= y[m]
нет
да
x[m] = y[m]
x[m] = y[m]
n = 1 - n
x[1 - m] = y[1 - m]
конец файла S[0]
да
нет
записываем в файл F[n] элемент x[0]
читаем из файла S[0] элемент y[0]
x[0] > y[0]
да
n = 1 - n
нет
x[0] = y[0]
конец файла S[1]
да
выход
нет
записываем в файл F[n] элемент x[1]
читаем из файла S[1] элемент y[1]
x[1] > y[1]
да
n = 1 - n
нет
x[1] = y[1]

стр

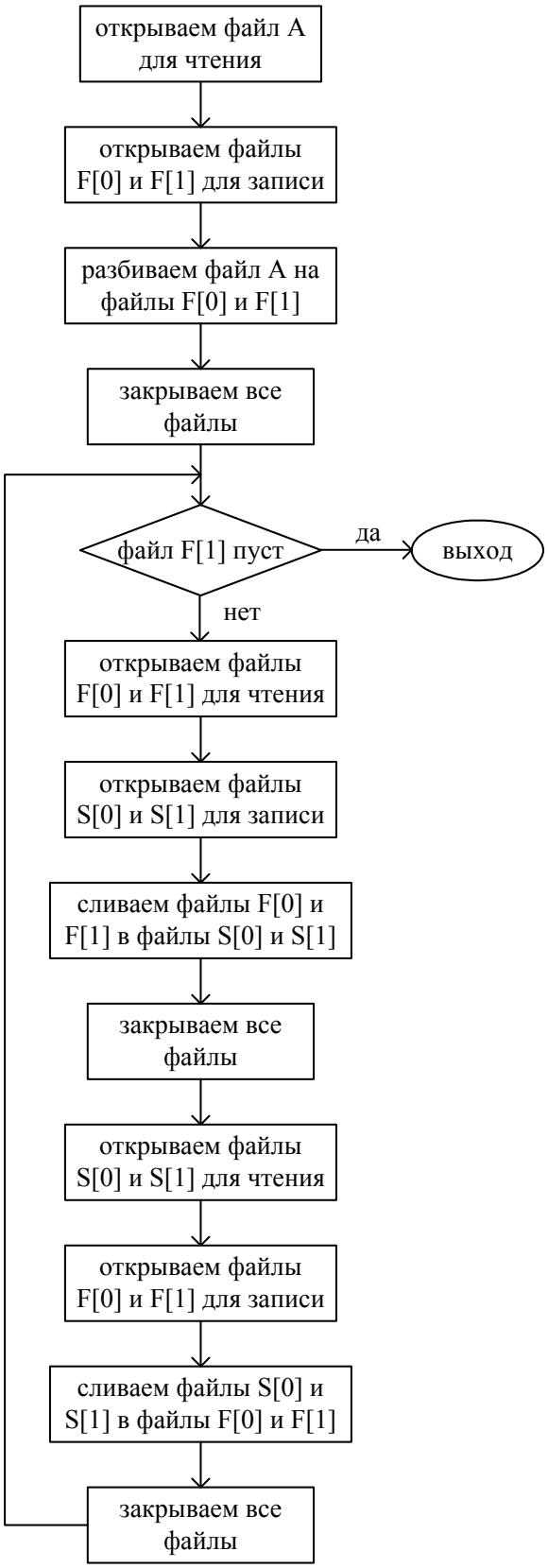
3.3.2.3. Сортировка естественным слиянием на четырех файлах

Непосредственно сортировка состоит из двух этапов: одного повторения фазы разбиения, и затем циклического повторения фазы слияния. Алгоритм можно изложить следующим образом.

Алгоритм сортировки естественным слиянием на четырех файлах

1. Открываем файл *A* для чтения.
2. Открываем файлы *F*[0] и *F*[1] для записи.
3. Разбиваем файл *A* на файлы *F*[0] и *F*[1].
4. Закрываем все файлы.
5. Если файл *F*[1] пуст, работа алгоритма закончена.
6. Открываем файлы *F*[0] и *F*[1] для чтения.
7. Открываем файлы *S*[0] и *S*[1] для записи.
8. Сливаем файлы *F*[0] и *F*[1] в файлы *S*[0] и *S*[1].
9. Закрываем все файлы.
10. Если файл *S*[1] пуст, работа алгоритма закончена.
11. Открываем файлы *S*[0] и *S*[1] для чтения.
12. Открываем файлы *F*[0] и *F*[1] для записи.
13. Сливаем файлы *S*[0] и *S*[1] в файлы *F*[0] и *F*[1].
14. Закрываем все файлы.
15. Повторяем алгоритм с шага 5.

Блок-схема сортировки естественным слиянием на четырех файлах



стр

3.3.3. Естественное слияние на 2N файлах

Идею алгоритма естественного слияния на четырех файлах можно распространить на любое четное (большее двух) количество файлов.

1. Файл *A* разбиваем на *N* файлов: *F*[0], *F*[1],…,*F*[*N*–1]. В файл *F*[0] записываем первый упорядоченный отрезок из файла *A*, в файл *F*[1] – второй, и т.д. пока не запишем

первые N отрезков. Затем снова записываем по одному отрезку в файл $F[0]$, $F[1]$ и т.д. пока не закончится файл A.

2. Если файл $F[1]$ не пуст, сливаем файлы $F[0]$, $F[1]$,…, $F[N–1]$ в файлы $S[0]$, $S[1]$,…, $S[N–1]$, объединяя первые упорядоченные отрезки из $F[0]$, $F[1]$,…, $F[N–1]$ в упорядоченный отрезок в файле $S[0]$, вторые упорядоченные отрезки – в упорядоченный отрезок в файле $S[1]$, и т.д.

3. Если файл $S[1]$ не пуст, сливаем файлы $S[0]$, $S[1]$,…, $S[N–1]$ в файлы $F[0]$, $F[1]$,…, $F[N–1]$, объединяя первые упорядоченные отрезки из $S[0]$, $S[1]$,…, $S[N–1]$ в упорядоченный отрезок в файле $S[0]$, вторые упорядоченные отрезки – в упорядоченный отрезок в файле $S[1]$, и т.д.

4. Повторяем шаги 2–3, пока это возможно, т.е. пока файл $F[1]$ или $S[1]$ не окажется пустым.

Приведем пример работы алгоритма при $N = 3$. Рассмотрим файл A из предыдущих примеров. Упорядоченные отрезки, которые войдут в разные файлы, выделены разными цветами.

A:	21	12	14	34	18	53	29	36	48	19	43

Разбиваем файл A на три части: $F[0]$, $F[1]$, $F[2]$.

$F[0]$:	21	29	36	48	
$F[1]$:	12	14	34	19	43
$F[2]$:	18	53			

стр

Теперь сливаем полученные последовательности в файлы $S[0]$, $S[1]$, $S[2]$, объединяя упорядоченные отрезки (выделенные разными цветами) в упорядоченные отрезки большей длины. Обратим внимание, что в итоге файл $S[2]$ оказался пустым, т.к. в файлах $F[0]$ и $F[2]$ было по одному упорядоченному отрезку, а в файле $F[1]$ – два, то три первых отрезка мы объединили в упорядоченный отрезок в файле $S[0]$, а второй отрезок из файла $F[1]$ полностью переписали в файл $S[1]$.

$S[0]$:	12	14	18	21	29	34	36	48	53
$S[1]$:	19	43							
$S[2]$:									

Аналогично сливаем файлы $S[0]$, $S[1]$, $S[2]$ в файлы $F[0]$, $F[1]$, $F[2]$.

$F[0]$:	12	14	18	19	21	29	34	36	43	48	53
$F[1]$:											
$F[2]$:											

Файл $F[1]$ оказался пустым, значит, работа алгоритма закончена. Упорядоченная последовательность содержится в файле $F[0]$.

Сортировка выполнены за два прохода, а не за три, как в случае естественного слияния на трех или четырех файлах. Платой за уменьшение числа проходов является увеличение числа необходимых файлов, т.е. выигрывая в скорости, мы проигрываем в количестве необходимой памяти.

стр

3.3.3.1. Фаза разбиения

Фаза разбиения сортировки естественным слиянием на $2N$ файлах практически ничем не отличается от фазы разбиения сортировки естественным слиянием на четырех файлах

(ссылка). Изменения в блок-схеме и алгоритме касаются лишь выбора n – номера файла, в который записывается текущий отрезок. Индекс n меняется от 0 до $N - 1$. Изменение индекса происходит каждый раз, когда заканчивается очередной упорядоченный отрезок, т.е. когда $x_1 > x_2$. Когда индекс n становится равным N, он полагается равным нулю (можно также использовать операцию «остаток от целочисленного деления»).

Приведем пример работы алгоритма на фазе разбиения при $N = 3$. Заголовок файла, в который идет запись, выделен зеленым цветом, в начале работы алгоритма $n = 0$. Элемент x_1 обведен в рамку, упорядоченные отрезки выделены разными цветами. Сначала считываем первый элемент из файла A в переменную x_1.

	x_1										
A:	21	12	14	34	18	53	29	36	48	19	43

Так как файл A не закончился, записываем элемент $x_1 = 12$ в файл $F[0]$ и считываем следующий элемент из файла A в переменную x_2.

$F[0]$: 21
$F[1]$:
$F[2]$:

	x_1	x_2								
A:	21	12	14	34	18	53	29	36	48	19	43

стр

Файл не A закончился, но $x_1 > x_2$, это означает, что первый упорядоченный отрезок закончился. Поэтому полагаем $x_1 = x_2$, изменяем значение индекса $n = n + 1 = 1$ и записываем второй упорядоченный отрезок в файл $F[1]$.

		$x_1 = x_2$									
A:	21	12	14	34	18	53	29	36	48	19	43

$F[0]$: 21
$F[1]$: 12
$F[2]$:

		x_1	x_2								
A:	21	12	14	34	18	53	29	36	48	19	43

			$x_1 = x_2$								
A:	21	12	14	34	18	53	29	36	48	19	43

$F[0]$: 21
$F[1]$: 12 14
$F[2]$:

			x_1	x_2							
A:	21	12	14	34	18	53	29	36	48	19	43

$x_1 = x_2$

A: 21 12 14 [34] 18 53 29 36 48 19 43

F[0]: 21
F[1]: 12 14 34
F[2]:

x_1 x_2

A: 21 12 14 [34] 18 53 29 36 48 19 43

стр

Файл не *A* закончился, но $x_1 > x_2$, это означает, что второй упорядоченный отрезок закончился. Поэтому полагаем $x_1 = x_2$, изменяем значение индекса $n = n + 1 = 2$ и записываем третий упорядоченный отрезок в файл *F*[2].

$x_1 = x_2$

A: 21 12 14 34 [18] 53 29 36 48 19 43

F[0]: 21
F[1]: 12 14 34
F[2]: 18

x_1 x_2

A: 21 12 14 34 [18] 53 29 36 48 19 43

$x_1 = x_2$

A: 21 12 14 34 18 [53] 29 36 48 19 43

F[0]: 21
F[1]: 12 14 34
F[2]: 18 53

x_1 x_2

A: 21 12 14 34 18 [53] 29 36 48 19 43

стр

Файл не *A* закончился, но $x_1 > x_2$, это означает, что третий упорядоченный отрезок закончился. Поэтому полагаем $x_1 = x_2$, изменяем значение индекса $n = n + 1 = 3$. Так как на этот раз $n = N$, полагаем $n = 0$ и записываем четвертый упорядоченный отрезок в файл *F*[0].

$x_1 = x_2$

A: 21 12 14 34 18 [53] 29 36 48 19 43

F[0]: 21 29
F[1]: 12 14 34
F[2]: 18 53

							x_1	x_2				
A:	21	12	14	34	18	53	29	36	48	19	43	

								$x_1 = x_2$				
A:	21	12	14	34	18	53	29	36	48	19	43	

F[0]:	21	29	36
F[1]:	12	14	34
F[2]:	18	53	

								x_1	x_2			
A:	21	12	14	34	18	53	29	36	48	19	43	

									$x_1 = x_2$			
A:	21	12	14	34	18	53	29	36	48	19	43	

F[0]:	21	29	36	48
F[1]:	12	14	34	
F[2]:	18	53		

								x_1	x_2			
A:	21	12	14	34	18	53	29	36	48	19	43	

Файл не *A* закончился, но $x_1 > x_2$, это означает, что четвертый упорядоченный отрезок закончился. Поэтому полагаем $x_1 = x_2$, изменяем значение индекса $n = n + 1 = 1$ и записываем пятый упорядоченный отрезок в файл *F*[1].

										$x_1 = x_2$		
A:	21	12	14	34	18	53	29	36	48	19	43	

F[0]:	21	29	36	48	
F[1]:	12	14	34	19	
F[2]:	18	53			

									x_1	x_2		
A:	21	12	14	34	18	53	29	36	48	19	43	

											$x_1 = x_2$	
A:	21	12	14	34	18	53	29	36	48	19	43	

F[0]:	21	29	36	48		
F[1]:	12	14	34	19	43	
F[2]:	18	53				

										x_1	x_2
A:	21	12	14	34	18	53	29	36	48	19	43	

Файл *A* закончился, работа алгоритма завершена.

стр

3.3.3.2. Фаза слияния

Рассмотрим отличие фазы слияния от фазы слияния сортировки естественным слиянием на четырех файлах (ссылка). Они касаются только выбора индексов: n – номера файла, в который записывается текущий отрезок, и m – номера файла, элемент из которого необходимо записать в файл $F[n]$. Индекс n меняется так же, как и на фазе разбиения (ссылка). Индекс m меняется несколько сложнее: это индекс минимального элемента из тех элементов $x[0],\ldots, x[N-1]$, где не закончился файл или упорядоченный отрезок. Для выбора минимума удобно завести массив переменных $flag[N]$, где переменная $flag[i]$ равна нулю, если закончился файл $F[i]$ или упорядоченный отрезок в этом файле, и единице в противном случае.

Итак, нам потребуются следующие переменные:

$S[N]$ – массив из N файлов, из которых производится чтение упорядоченных отрезков;

$F[N]$ – массив из N файлов, в которые производится запись объединенных упорядоченных отрезков;

$x[N]$ – массив из текущих элементов, считанных из файлов $S[0], S[1],\ldots, S[N-1]$;

$y[N]$ – массив из следующих элементов файлов $S[0], S[1],\ldots, S[N-1]$;

$flag[N]$ – массив признаков конца файла или отрезка;

n – номер файла, в который записывается текущий отрезок;

m – номер файла, элемент из которого необходимо записать в файл $F[n]$.

Приведем пример слияния двух файлов. Упорядоченные отрезки, подлежащие слиянию, выделены разными цветами. Считываем сначала элементы $x[0], x[1],\ldots, x[N-1]$ из файлов $S[0], S[1],\ldots, S[N-1]$ соответственно. Полагаем $n = 0$, $m = 0$. Это означает, что сливать отрезки мы будем в файл $F[0]$, а записывать очередной элемент будем из файла $S[0]$. Но если окажется, что наименьший элемент находится не в файле $S[0]$, нужно будет изменить индекс. Будем выделять заголовки файлов $S[m]$ и $F[n]$ зеленым цветом. Из элементов $x[0], x[1],\ldots, x[N-1]$ обведены в рамку только те, из которых на данном шаге выбирается минимальный элемент (т.е. если не закончился соответствующий файл или отрезок). Так как в данном примере все файлы изначально не пусты, то все $flag[i] = 1$, и все $x[i]$ обведены в рамку.

			$x[0]$				
$flag[0] = 1;$	$m = 0;$	$S[0]:$	21	29	36	48	
			$x[1]$				
$flag[1] = 1;$		$S[1]:$	12	14	34	19	43
			$x[2]$				
$flag[2] = 1;$		$S[2]:$	18	53			

Минимальный элемент среди выделенных – это $x[1]$, значит, полагаем $m = 1$, записываем элемент $x[m] = 12$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[1]$ элемент $y[m]$.

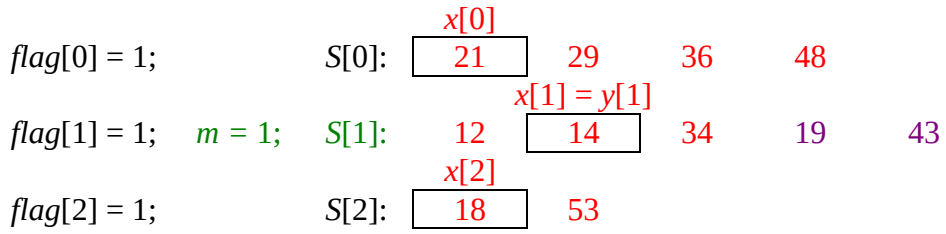
			$x[0]$				
$flag[0] = 1;$		$S[0]:$	21	29	36	48	
			$x[1]$	$y[1]$			
$flag[1] = 1;$	$m = 1;$	$S[1]:$	12	14	34	19	43
			$x[2]$				
$flag[2] = 1;$		$S[2]:$	18	53			

$n = 0$; $F[0]$: 12

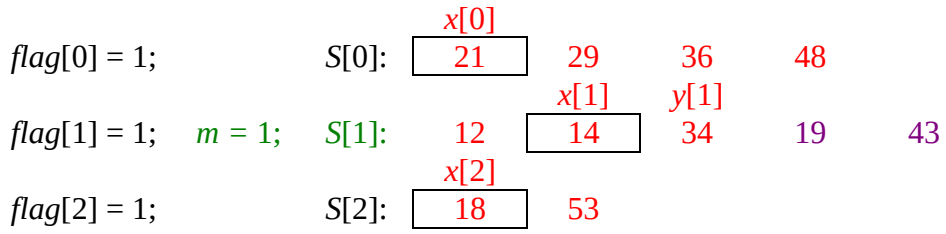
$F[1]$:

$F[2]$:

Файл $S[m] = S[1]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.



Минимальный элемент среди выделенных – это $x[1]$, значит, полагаем $m = 1$, записываем элемент $x[m] = 14$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[1]$ элемент $y[m]$.

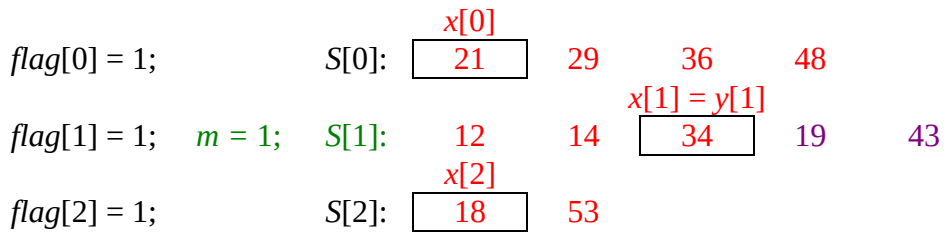


$n = 0$; $F[0]$: 12 14

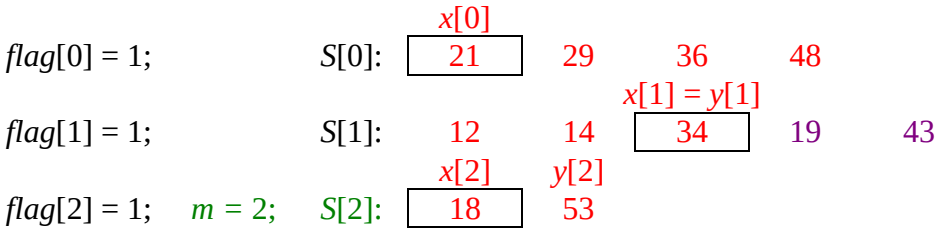
$F[1]$:

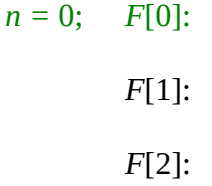
$F[2]$:

Файл $S[m] = S[1]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.

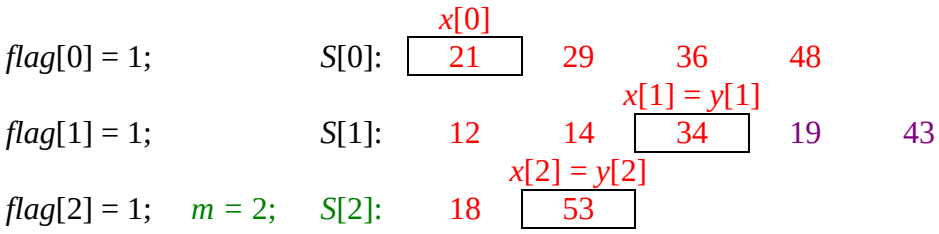


Минимальный элемент среди выделенных – это $x[2]$, значит, полагаем $m = 2$, записываем элемент $x[m] = 18$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[2]$ элемент $y[m]$.

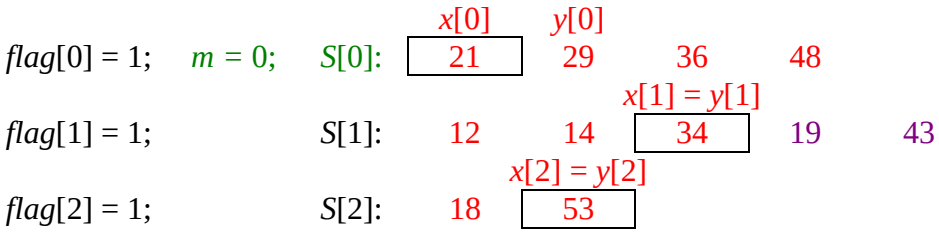


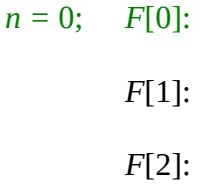


Файл $S[m] = S[2]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.

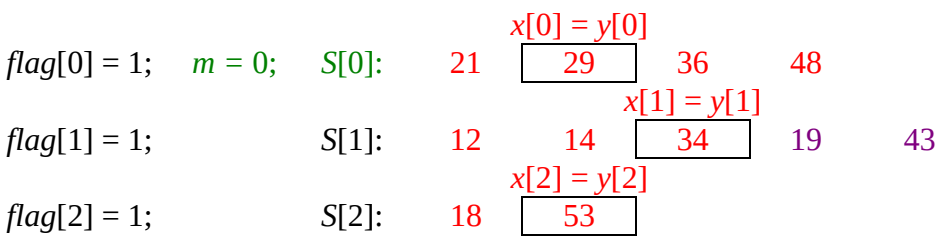


Минимальный элемент среди выделенных – это $x[0]$, значит, полагаем $m = 0$, записываем элемент $x[m] = 21$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[0]$ элемент $y[m]$.

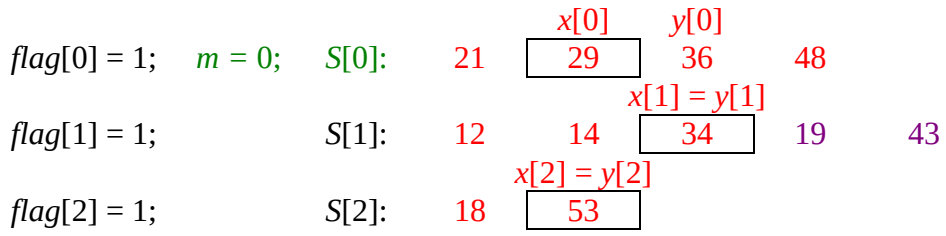


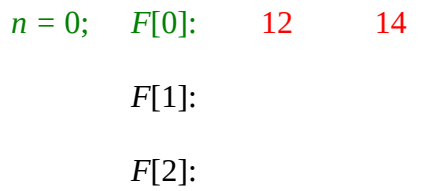


Файл $S[m] = S[0]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.

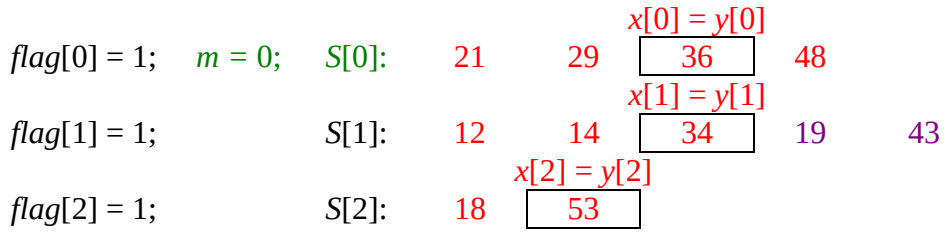


Минимальный элемент среди выделенных – это $x[0]$, значит, полагаем $m = 0$, записываем элемент $x[m] = 29$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[0]$ элемент $y[m]$.

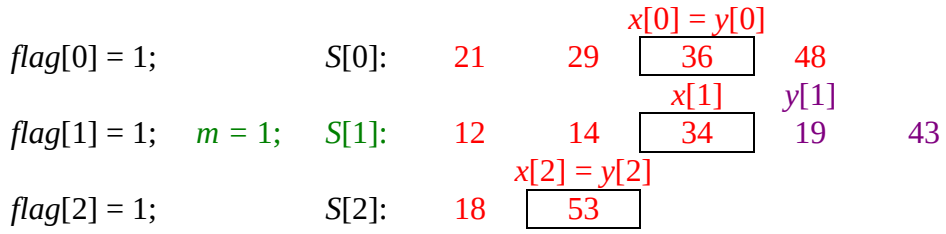


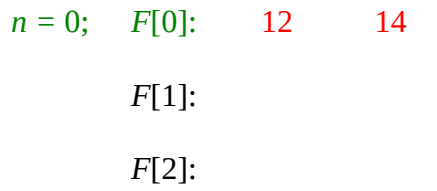


Файл $S[m] = S[0]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.



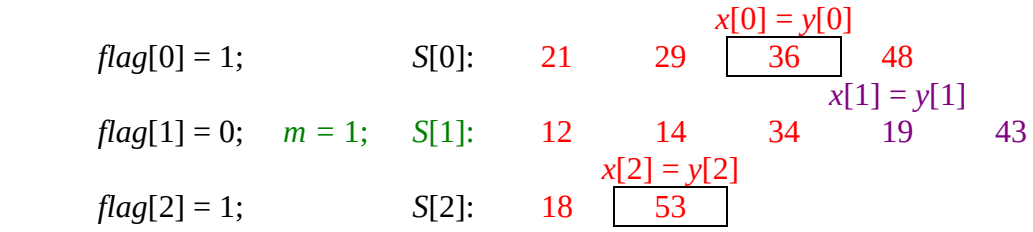
Минимальный элемент среди выделенных – это $x[1]$, значит, полагаем $m = 1$, записываем элемент $x[m] = 34$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[1]$ элемент $y[m]$.



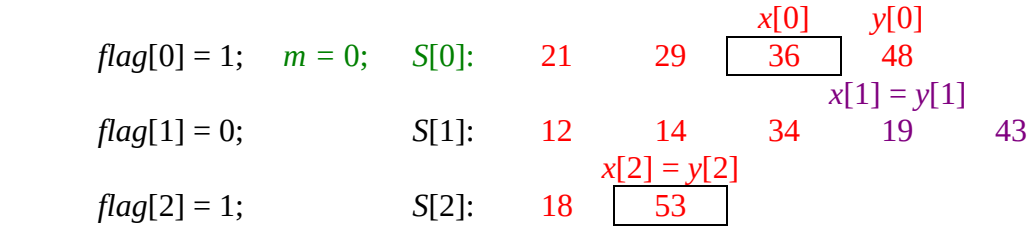


Файл $S[m] = S[1]$ не закончился, но $x[m] > y[m]$, это означает, что закончился упорядоченный отрезок в файле $S[m]$. Поэтому полагаем $flag[m] = 0$ и $x[m] = y[m]$. Повторяем те же действия, но далее элемент $x[1]$ не будет участвовать в выборе минимума, пока не закончатся упорядоченные отрезки в остальных файлах.

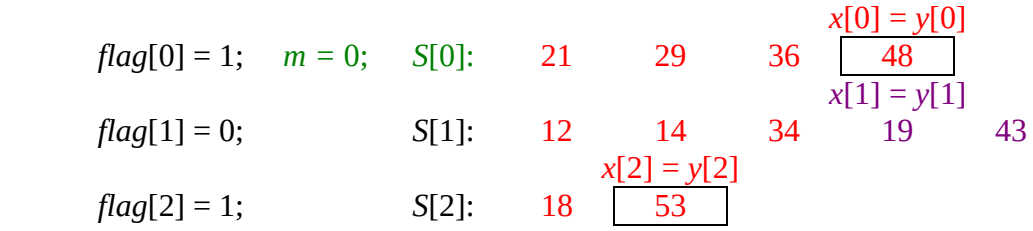
стр



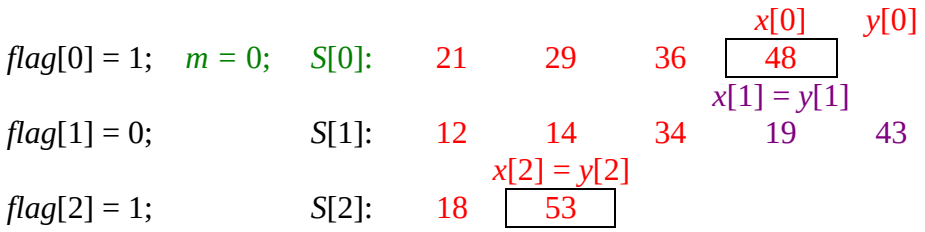
Минимальный элемент среди выделенных – это $x[0]$, значит, полагаем $m = 0$, записываем элемент $x[m] = 36$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[0]$ элемент $y[m]$.



Файл $S[m] = S[0]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.



Минимальный элемент среди выделенных – это $x[0]$, значит, полагаем $m = 0$, записываем элемент $x[m] = 48$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[0]$ элемент $y[m]$.



$n = 0$; $F[0]$: 12 14 18 21 29 34 36 48

$F[1]$:

$F[2]$:

стр

Файл $S[m] = S[0]$ закончился, поэтому полагаем $flag[m] = 0$. Повторяем те же действия, но далее элемент $x[0]$ не будет участвовать в выборе минимума, пока не закончатся упорядоченные отрезки в остальных файлах.

							$x[0]$	$y[0]$
$flag[0] = 0$;	$m = 0$;	$S[0]$:	21	29	36	48		
						$x[1] = y[1]$		
$flag[1] = 0$;		$S[1]$:	12	14	34	19	43	
				$x[2] = y[2]$				
$flag[2] = 1$;		$S[2]$:	18	53				

Минимальный элемент среди выделенных – это $x[2]$ (собственно, остался только один выделенный элемент), значит, полагаем $m = 2$, записываем элемент $x[m] = 53$ в файл $F[n] = F[0]$ и читаем из файла $S[m] = S[0]$ элемент $y[m]$.

							$x[0]$	$y[0]$
$flag[0] = 0$;		$S[0]$:	21	29	36	48		
						$x[1] = y[1]$		
$flag[1] = 0$;		$S[1]$:	12	14	34	19	43	
				$x[2]$	$y[2]$			
$flag[2] = 1$;	$m = 2$;	$S[2]$:	18	53				

$n = 0$; $F[0]$: 12 14 18 21 29 34 36 48 53

$F[1]$:

$F[2]$:

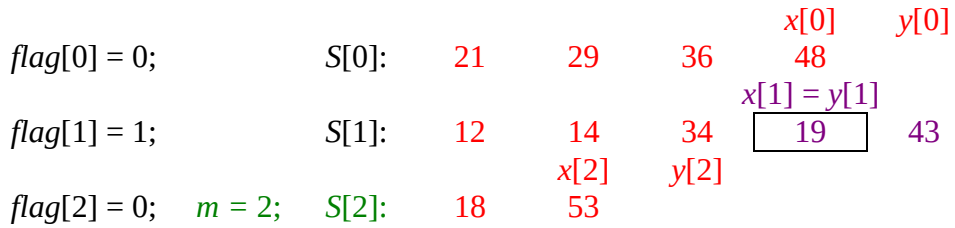
Файл $S[m] = S[2]$ закончился, поэтому полагаем $flag[m] = 0$. Повторяем те же действия, но далее элемент $x[2]$ не будет участвовать в выборе минимума, пока не закончатся упорядоченные отрезки в остальных файлах.

							$x[0]$	$y[0]$
$flag[0] = 0$;		$S[0]$:	21	29	36	48		
						$x[1] = y[1]$		
$flag[1] = 0$;		$S[1]$:	12	14	34	19	43	
				$x[2]$	$y[2]$			
$flag[2] = 0$;	$m = 2$;	$S[2]$:	18	53				

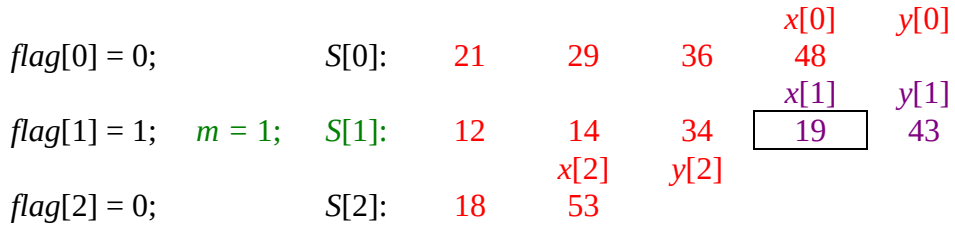
стр

Здесь все переменные $flag[i] = 0$, это означает, что во всех файлах упорядоченные отрезки закончились. Поэтому полагаем $n = n + 1 = 1$ и начинаем слияние очередных

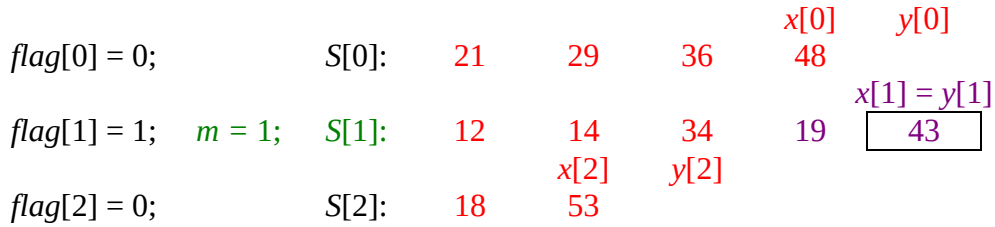
упорядоченных отрезков в файл $F[n] = F[1]$. Так как файл $S[1]$ не закончился, полагаем $flag[1] = 1$. Остальные файлы закончились, поэтому $flag[0] = flag[2] = 0$.



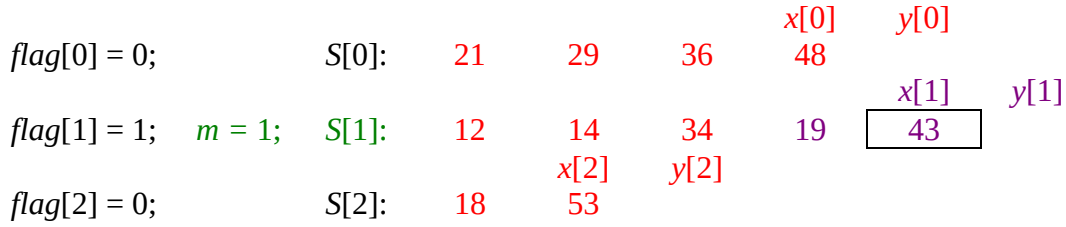
Минимальный элемент среди выделенных – это $x[1]$ (собственно, остался только один выделенный элемент), значит, полагаем $m = 1$, записываем элемент $x[m] = 19$ в файл $F[n] = F[1]$ и читаем из файла $S[m] = S[1]$ элемент $y[m]$.



Файл $S[m] = S[1]$ не закончился, и $x[m] < y[m]$, это означает, что упорядоченный отрезок также не закончился. Поэтому полагаем $x[m] = y[m]$ и повторяем те же действия.



Минимальный элемент среди выделенных – это $x[1]$, значит, полагаем $m = 1$, записываем элемент $x[m] = 43$ в файл $F[n] = F[1]$ и читаем из файла $S[m] = S[1]$ элемент $y[m]$.



	F[0]:	12	14	18	21	29	34	36	48	53
n = 1;	*F*[1]:	19	43							
	F[2]:									

Файл *S*[*m*] = *S*[1] закончился, поэтому полагаем *flag*[*m*] = 0. Повторяем те же действия, но далее элемент *x*[1] не будет участвовать в выборе минимума, пока не закончатся упорядоченные отрезки в остальных файлах.

							x[0]	*y*[0]	
flag[0] = 0;		*S*[0]:	21	29	36	48			
							x[1]	*y*[1]	
flag[1] = 0;	*m* = 1;	*S*[1]:	12	14	34	19	43		
				x[2]	*y*[2]				
flag[2] = 0;		*S*[2]:	18	53					

Здесь все переменные *flag*[*i*] = 0, это означает, что во всех файлах упорядоченные отрезки закончились. Поэтому полагаем *n* = *n* + 1 = 2 и начинаем слияние очередных упорядоченных отрезков в файл *F*[*n*] = *F*[2]. Но все файлы закончились, поэтому в данном случае *flag*[0] = *flag*[1] = *flag*[2] = 0, и работа алгоритма завершена. Обратим внимание, что мы не записали ни одного элемента в файл *F*[2], так как упорядоченные отрезки закончились раньше.

Приведем алгоритм фазы слияния. Дополнительно нам потребуется две переменные:

NF – число незакончившихся файлов *S*[0], *S*[1],…, *S*[*N*–1];

NO – число незакончившихся сливаемых упорядоченных отрезков.

стр

Алгоритм фазы слияния

1. Полагаем *i* = 0, *NF* = 0.
2. Пока *i* < *N*, делаем:
   2.1. читаем элемент *x*[*i*] из файла *S*[*i*];
   2.2. если не конец файла *S*[*i*], делаем:
      2.2.1. полагаем *flag*[*i*] = 1;
      2.2.2. полагаем *NF* = *NF* + 1;
   2.3. иначе полагаем *flag*[*i*] = 0;
   2.4. полагаем *i* = *i* + 1.
3. Полагаем *n* = 0, *m* = 0.
4. Пока *NF* > 0, делаем:
   4.1. полагаем *NO* = *NF*;
   4.2. пока *NO* > 0, делаем:
      4.2.1. полагаем *i* = 0, *min* = 32765;
      4.2.2. пока *i* < *N*, делаем:
         4.2.2.1. если *flag*[*i*] = 1 и *x*[*i*] < *min*, полагаем *min* = *x*[*i*], *m* = *i*;
         4.2.2.2. полагаем *i* = *i* + 1;
      4.2.3. записываем элемент *x*[*m*] в файл *F*[*n*];
      4.2.4. читаем элемент *y*[*m*] из файла *S*[*m*];
      4.2.5. если конец файла *S*[*m*], делаем:
         4.2.5.1. полагаем *flag*[*m*] = 0;
         4.2.5.2. полагаем *NF* = *NF* – 1;

4.2.5.3. полагаем $NO = NO - 1$;
4.2.6. иначе если $x[m] > y[m]$, делаем:
4.2.6.1. полагаем $flag[m] = 0$;
4.2.6.2. полагаем $NO = NO - 1$;
4.2.7. полагаем $x[m] = y[m]$;
4.3. полагаем $n = n + 1$;
4.4. если $n = N$, полагаем $n = 0$;
4.5. полагаем $i = 0$;
4.6. пока $i < N$, делаем:
4.6.1. если не конец файла $S[i]$, полагаем $flag[i] = 1$;
4.6.2. полагаем $i = i + 1$.

стр.

3.3.3.3. Сортировка естественным слиянием на 2N файлах

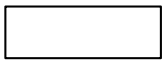
Непосредственно сортировка состоит из двух этапов: одного повторения фазы разбиения, и затем циклического повторения фазы слияния. Алгоритм можно изложить следующим образом.

Алгоритм естественного слияния на 2*N* файлах

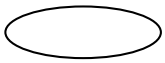
1. Открываем файл A для чтения.
2. Открываем файлы $F[0], F[1],\ldots, F[N-1]$ для записи.
3. Разбиваем файл A на файлы $F[0], F[1],\ldots, F[N-1]$.
4. Закрываем все файлы.
5. Если файл $F[1]$ не пуст выполняем:
6. открываем файлы $F[0], F[1],\ldots, F[N-1]$ для чтения;
7. открываем файл $S[0], S[1],\ldots, S[N-1]$ для записи;
8. сливаем файлы $F[0], F[1],\ldots, F[N-1]$ в файлы $S[0], S[1],\ldots, S[N-1]$;
9. закрываем все файлы;
10. Если файл $S[1]$ не пуст выполняем:
11. открываем файлы $S[0], S[1],\ldots, S[N-1]$ для чтения;
12. открываем файлы $F[0], F[1],\ldots, F[N-1]$ для записи;
13. сливаем файлы $S[0], S[1],\ldots, S[N-1]$ в файлы $F[0], F[1],\ldots, F[N-1]$;
14. закрываем все файлы;
15. повторяем шаги с 5 по 14.

стр

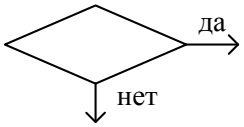
Основные элементы блок-схемы.



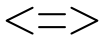
блок действия, команды



блок вывода результата



блок условия
если условие истинно, идти
по ветке "да", в противном
случае, по ветке "нет"



Обмен значениями

стр

ОГЛАВЛЕНИЕ