УДК 004.021

О ВЫДЕЛЕНИИ МАКРООПЕРАЦИЙ ИЗ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ СОРТИРОВКИ МАССИВОВ ДАННЫХ

О.Н. Паулин, Н.О. Комлевая, С.Ю. Марулин

Розглядаються обчислювальні процеси (ОП) як простих, так і ефективних сортувань. Для наочності ОП представляються схемами алгоритмів. У кожному процесі виділяються макрооперації (МО), які є функціонально закінченими фрагментами ОП. Списки МО для кожної сортування зводяться воєдино і узагальнюються, тобто зводяться до мінімуму МО. Це дозволяє розглядати більшість ОП сортувань в рамках виділених МО.
Ключові слова: сортування, обчислювальний процес, схема алгоритму, макрооперація.

Рассматриваются вычислительные процессы (ВП) как простых, так и эффективных сортировок. Для наглядности ВП представляются схемами алгоритмов. В каждом процессе выделяются макрооперации (МО), которые являются функционально законченными фрагментами ВП. Списки МО для каждой сортировки сводятся воедино и обобщаются, то есть сводятся к минимальному количеству МО. Это позволяет рассматривать большинство ВП сортировок в рамках выделенных МО.
Ключевые слова: сортировка, вычислительный процесс, схема алгоритма, макрооперации.

The paper dwells computation process (CP) array sorting from the simple to the efficient. Computation process present's as a block flowchart for clarity. Each process are extracted macro operations (MO), which are functionality finished pieces of CP. Lists of MO for each sorting algorithm put it together and summarized – minimizing number of MO. This approach will allow careful consideration most CP as it applies to dedicated MO.
Key words: sorting algorithm, computation process, flowchart, macro operations.

Введение

В классической работе [1] рассмотрены различные сортировки массивов данных, проведен их анализ с точки зрения эффективности. Под *сортировкой* понимается процесс перестановки объектов некоторого множества в определенном порядке. Цель сортировки – облегчить последующий поиск объектов (элементов) в отсортированном множестве. Упорядоченные объекты содержатся в телефонных книгах, библиотеках, словарях, на складах и т.д., то есть всюду, где появляется необходимость в их поиске.

Под качеством программного обеспечения (ПО), согласно стандартам (ISO 8402:1994, 1061-1998 IEEE), понимается совокупность характеристик ПО, относящихся к его способности удовлетворять установленные и предполагаемые потребности конечного пользователя. Показатель качества включает различные метрики оценки, которые можно использовать для контроля разработки ПО на различных стадиях его жизненного цикла. В качестве оценок чаще всего используются *надёжность*, *сложность*, *безошибочность*. В работе речь идёт о безошибочности программ.

Удовлетворение требованиям к качеству программы, т.е. получение наилучших указанных оценок, достигается на стадии отладки программы. Часто процесс отладки программы занимает большее время, чем написание программы, что является большой проблемой, особенно на этапе ввода программы в эксплуатацию.

Одним из перспективных подходов к решению данной проблемы является представление программы в виде конечного автомата [2]. Достоинство автоматного подхода к программированию при анализе программ – относительная простота и наглядность отладки автомата, так как автомат является хорошо изученной моделью дискретного преобразования информации. Недостатком автоматного подхода, как впрочем, и других подходов к анализу программ, является запоздалое обнаружение ошибок в программе, что приводит к большим затратам времени на их исправление.

Нами предложено выявлять ошибки программ на более раннем этапе решения определённого класса задач, а именно на этапе их алгоритмизации, а также использовать для отладки программы сеть Петри [3], как модель с большей выразительностью языка, чем конечный автомат.

В [4] сделана первая попытка описания вычислительного процесса (ВП) простой сортировки вставками в терминах МО и моделирования этого ВП на сети Петри. Ниже анализируются избранные сортировки на предмет выявления МО, общих для класса ВП сортировок массивов.

Цель работы: снижение ошибок в программах сортировок массивов путём выделения МО из их вычислительных процессов и систематизации полученных МО.

Основная часть

Методы сортировки обычно разделяют на два класса: сортировка *массивов* и сортировка (последовательных) *файлов*. Часто их называют *внутренней* и *внешней* сортировкой, так как массивы располагаются во «внутренней» (оперативной) памяти ЭВМ, а файлы хранятся в более медленной, но более вместительной «внешней» памяти (диски). Ниже рассматривается только сортировки массивов.

Рассмотрим терминологию и обозначения, которые мы будем использовать в работе. Нам даны элементы a_1, a_2, …, a_n. Сортировка – это перестановка (в результате k шагов) этих элементов в таком порядке a_{k1}, a_{k2}, …, a_{kn}, что при заданной функции упорядочения f справедливо отношение $f(a_{k1}) \le f(a_{k2}) \le \ldots \le f(a_{kn})$.

Обычно функция упорядочения не вычисляется по какому-то специальному правилу, а содержится в каждом элементе в виде явной компоненты (поля). Её значение называют *ключом* элемента. Ключ в алгоритмах сортировки – единственная существенная компонента.

Эффективность алгоритма, реализующего тот или иной метод сортировки массивов, определяется двумя основными требованиями – экономичное использование памяти и быстродействие алгоритма. Первое требование означает, что переупорядочение элементов нужно выполнить *на том же месте памяти*; удобная мера быстродействия алгоритма получается при подсчёте числа C необходимых сравнений ключей и M пересылок элементов. Эти числа определяются некоторыми функциями от числа *n* сортируемых элементов.

Методы, сортирующие элементы на том же месте, можно разбить на 3 основных класса, в зависимости от лежащего в их основе принципа: сортировка вставками, сортировка выбором, сортировка обменом.

Определение. *Макрооперацией* мы будем называть функционально законченный фрагмент вычислительного процесса.

Выделение макроопераций из ВП обосновывается интуитивно понятным утверждением.

Утверждение. Большое множество ВП может быть построено на основе небольшого числа МО.

В применении к сортировкам это означает выделение МО из ВП как простых, так и эффективных сортировок, имея в виду в дальнейшем построение для каждой МО соответствующего фрагмента сети Петри, сборки фрагментов в общую сеть Петри для данного ВП и её моделирование на предмет получения безошибочной модели для конкретной сортировки.

Простые сортировки

В работе [4] подробно рассмотрена одна из простых сортировок, а именно: сортировка вставками (включениями), выделены МО и отлажена модель этой сортировки в виде ингибиторной сети Петри. В данной работе мы ограничимся только выделением и систематизацией МО.

По аналогии с [4] проведём анализ и других простых сортировок: простым выбором и простым обменом. Рассмотрим сортировки по возрастанию (основные идеи сохраняют свой смысл и при сортировке по убыванию). На рис. 1 показаны модели всех простых сортировок в виде числовой оси, на которой точками отмечена последовательность элементов массива. Последовательность разбита на *готовую* и *входную* подпоследовательности. Скобками показаны возможные перемещения элементов (как в сортировке вставками) или сравниваемые элементы.

Для моделей вводится базовое понятие «*граница*», разделяющая готовую и входную подпоследовательности. При этом используются уточнения: *текущая*, *следующая* и *начальная* границы. *Граничным элементом* является элемент, стоящий непосредственно за границей.

Во всех моделях рассматривается некое текущее состояние процесса сортировки и указывается новое положение границы, а также начальное, стартовое её положение. В понятие «модель» входит также способ организации процесса упорядочивания элементов массива, т.е. метод сортировки.

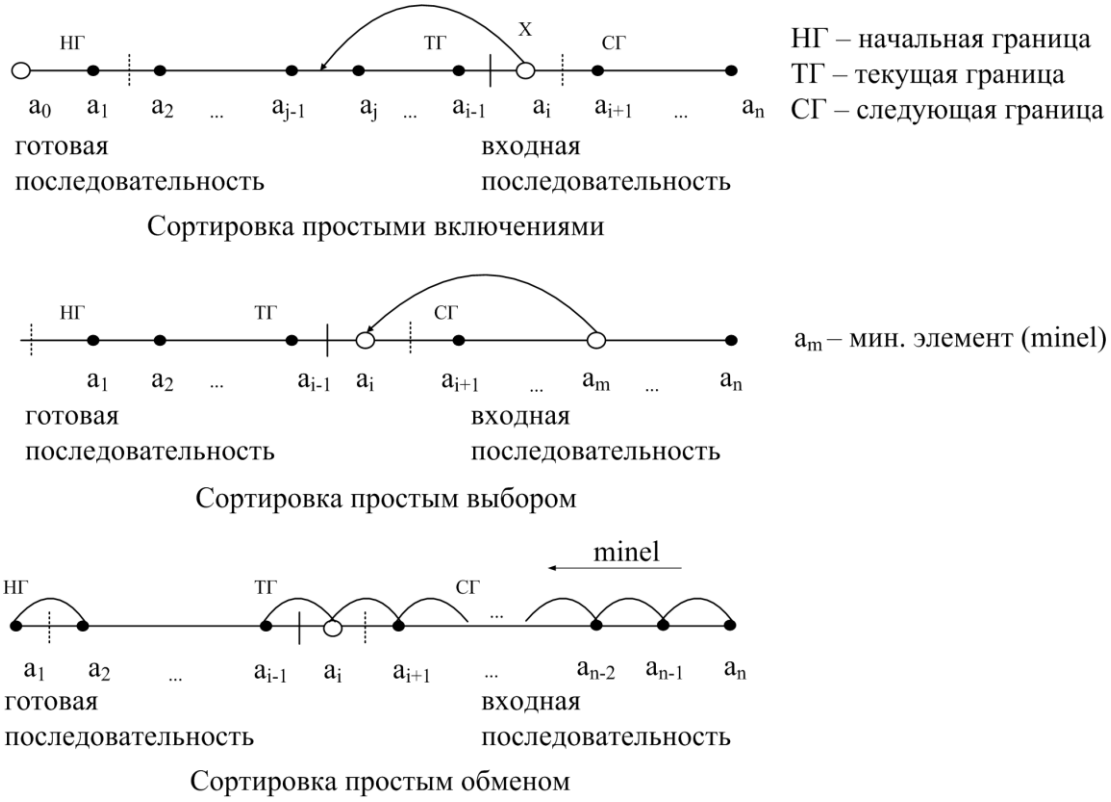

Рис. 1. Модели простых сортировок

В методе простых вставок для текущего граничного элемента x ищется *подходящее место* в готовой подпоследовательности, для чего используется двойное неравенство: $a_{j-1} \le x \le a_j$. Поскольку x может оказаться на первой позиции, а в этом случае отсутствует левая часть двойного неравенства, вводится фиктивный элемент a_0, называемый «*барьером*». Для гарантированного попадания x на первую позицию принимают $a_0 = x$.

В методе простого выбора ищется минимальный элемент a_m (minel) во входной подпоследовательности, который сравнивается с текущим граничным элементом a_i. Если он оказывается меньшим по значению граничного элемента, $a_m < a_i$, то они обмениваются местами.

В методе простого обмена (метод «пузырька») процесс протекает следующим образом. Сравнивается пара рядом стоящих элементов, начиная с конца последовательности. Если правый элемент пары оказывается меньше левого её элемента, $a_j < a_{j-1}$, то они обмениваются местами. Затем левый элемент уточнённой пары становится правым элементом новой пары элементов. Таким путём меньший по величине элемент продвигается влево, пока не встретит элемент с ещё меньшим значением. Тогда он останавливается перед ним и «передаёт ему эстафету». Элемент с наименьшим на данном проходе значением продвигается до граничного элемента. Ясно, что за один проход возможно перемещение нескольких элементов.

Во всех трёх методах после определения нового элемента готовой подпоследовательности граница передвигается вправо. Так проходит процесс до последнего элемента входной подпоследовательности. Стартовые положения границ для данных методов показаны на рисунке 1.

На рис. 2, 3, 4 подробно показаны схемы алгоритмов (СА), построенные в соответствии с рассмотренными моделями процессов сортировки. В частности, показаны счётчики итераций для внутренних и внешних циклов. Пунктиром во всех трёх схемах представлен внутренний цикл (тело внешнего цикла).

На основе СА проведен анализ процессов сортировки на предмет выделения МО для сортировок выбором и обменом (для сортировок вставками такой анализ выполнен в [4]).

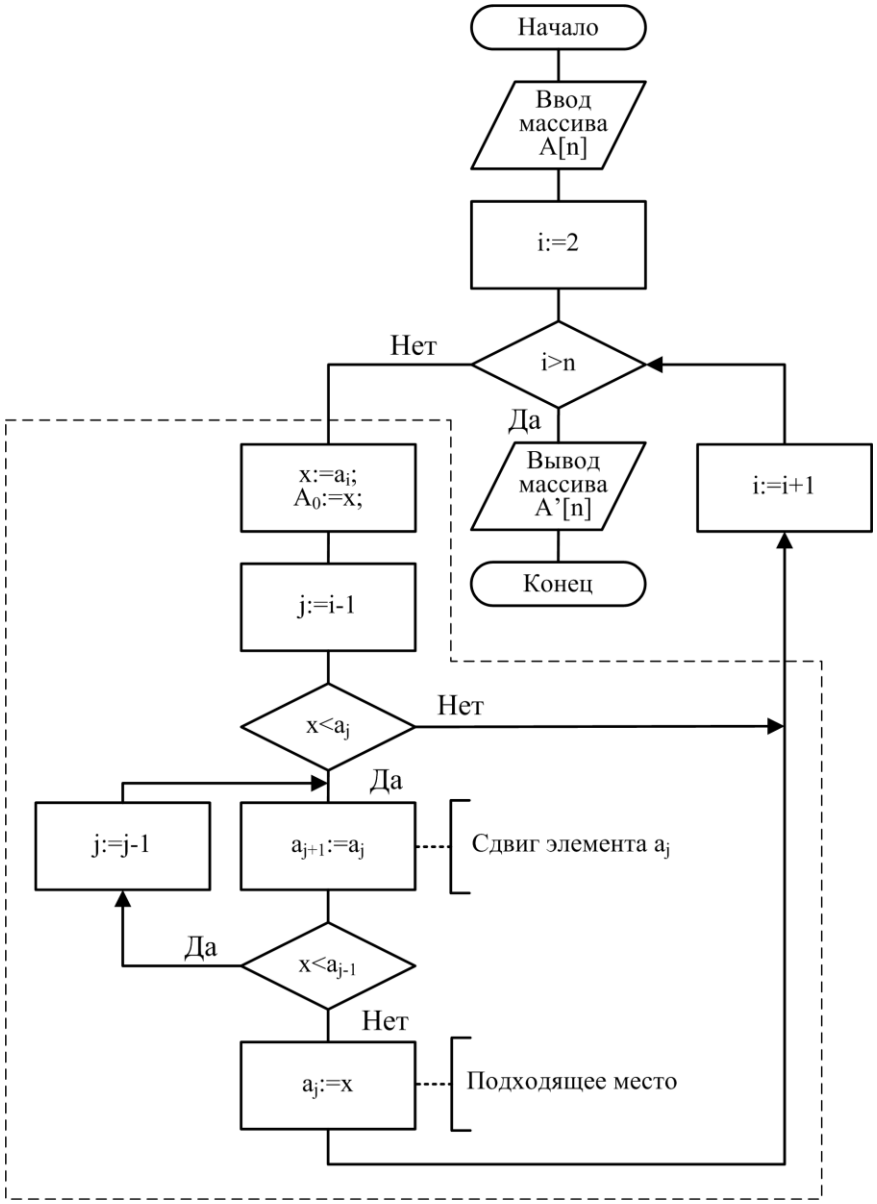

Рис. 2. Схема алгоритма простых вставок

Приведём список МО для СА простых вставок из [4]:

1) МО «Поиск подходящего места» (это совместная реализация МО «Счётный цикл» и МО «Сравнение по двойному неравенству»);

2) МО «Сдвиг элемента на одну позицию вправо».

Анализ СА простого выбора (рисунок 3) позволил выделить следующие МО:

1) МО «Нахождение минимального элемента в массиве»;

2) МО «Сравнить – переставить».

Анализ СА сортировки простым обменом показал, что ВП данной сортировки содержит:

1) МО «Формирование пары элементов»;

2) МО «Сравнить – переставить».

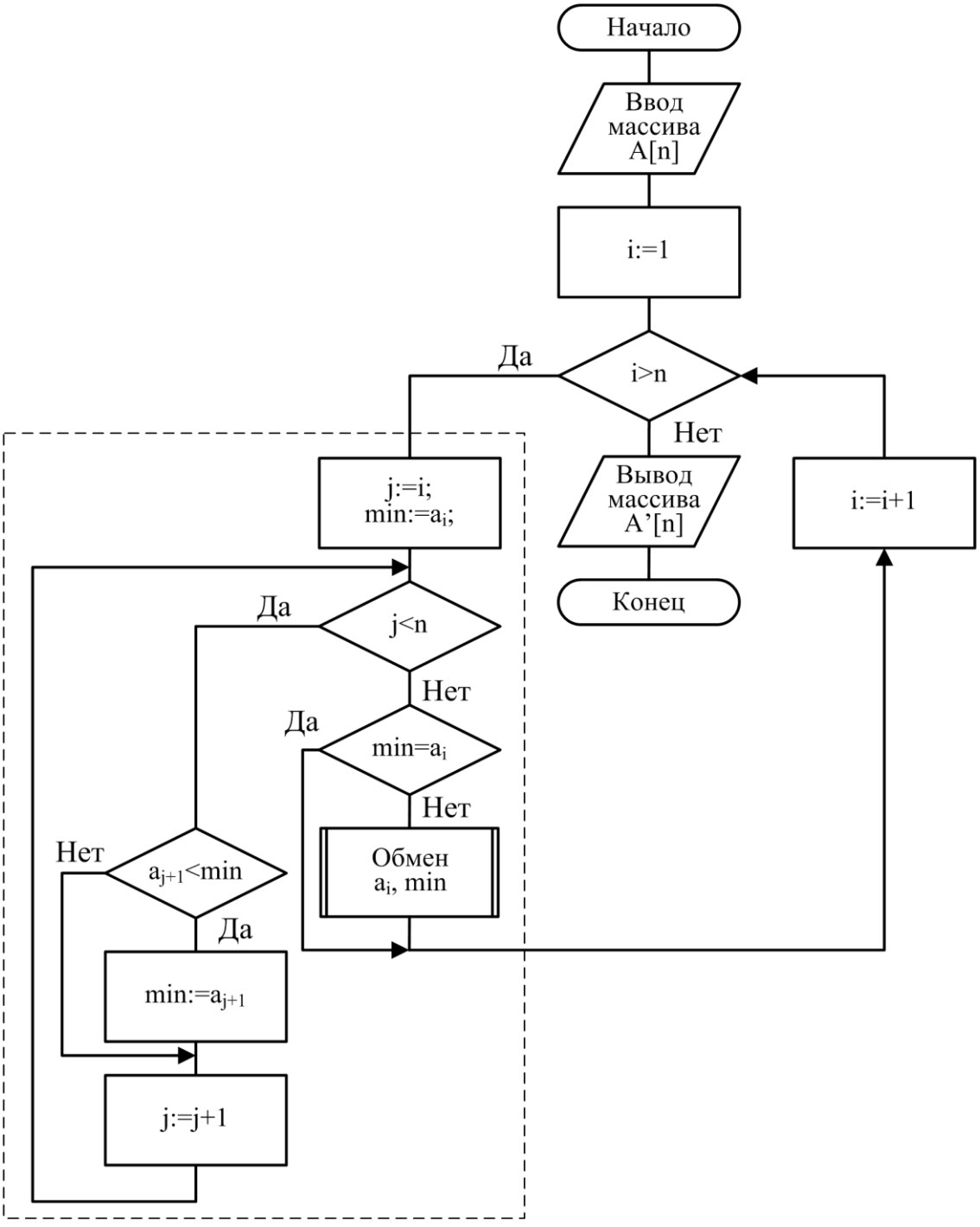

Рис. 3. Схема алгоритма для сортировки простым выбором

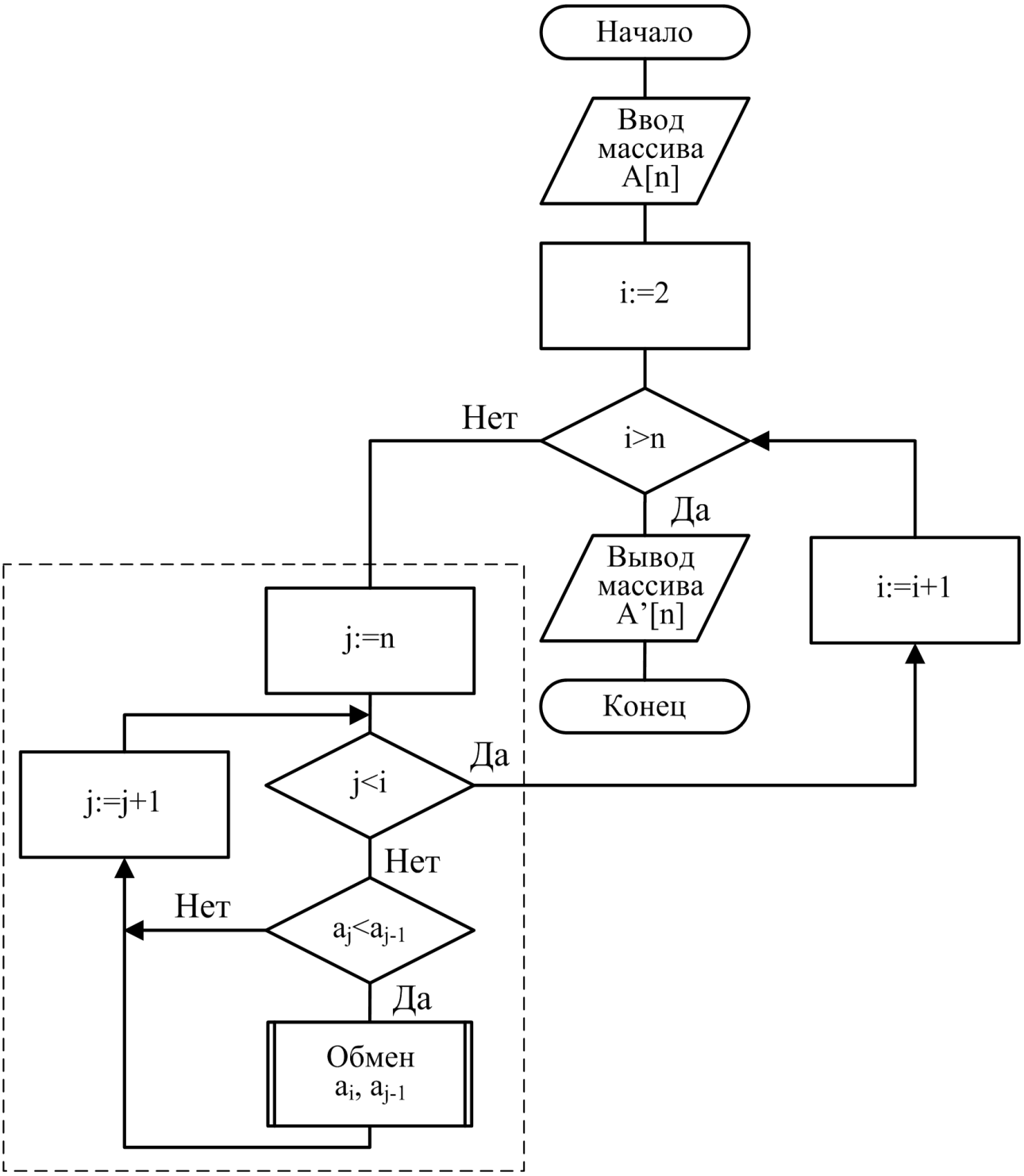

Рис. 4. Схема алгоритма сортировки простыми обменами

Из приведенных списков сортировок выделены общие для них МО: «Ввод массива», «Вывод массива», «Присваивание»; «Присваивание с вычислением», МО «Перемещение границы» (счётный цикл).

Эффективные сортировки

Рассмотрим более сложные сортировки: Шелла, быструю и пирамидальную (рис. 5, 6) [5]. Выбор этих сортировок связан с их популярностью. Они предназначены для обработки массивов большого и сверхбольшого объёма, которые невозможно обработать простыми методами за приемлемое время, так как сложность простых сортировок определяется как $O(n^2)$, в то время как упомянутые сложные сортировки в среднем имеют сложность $O(n\log_2 n)$, причём с увеличением n преимущество сложных сортировок быстро возрастает.

При построении СА для эффективных сортировок мы используем обобщённое представление блоков.

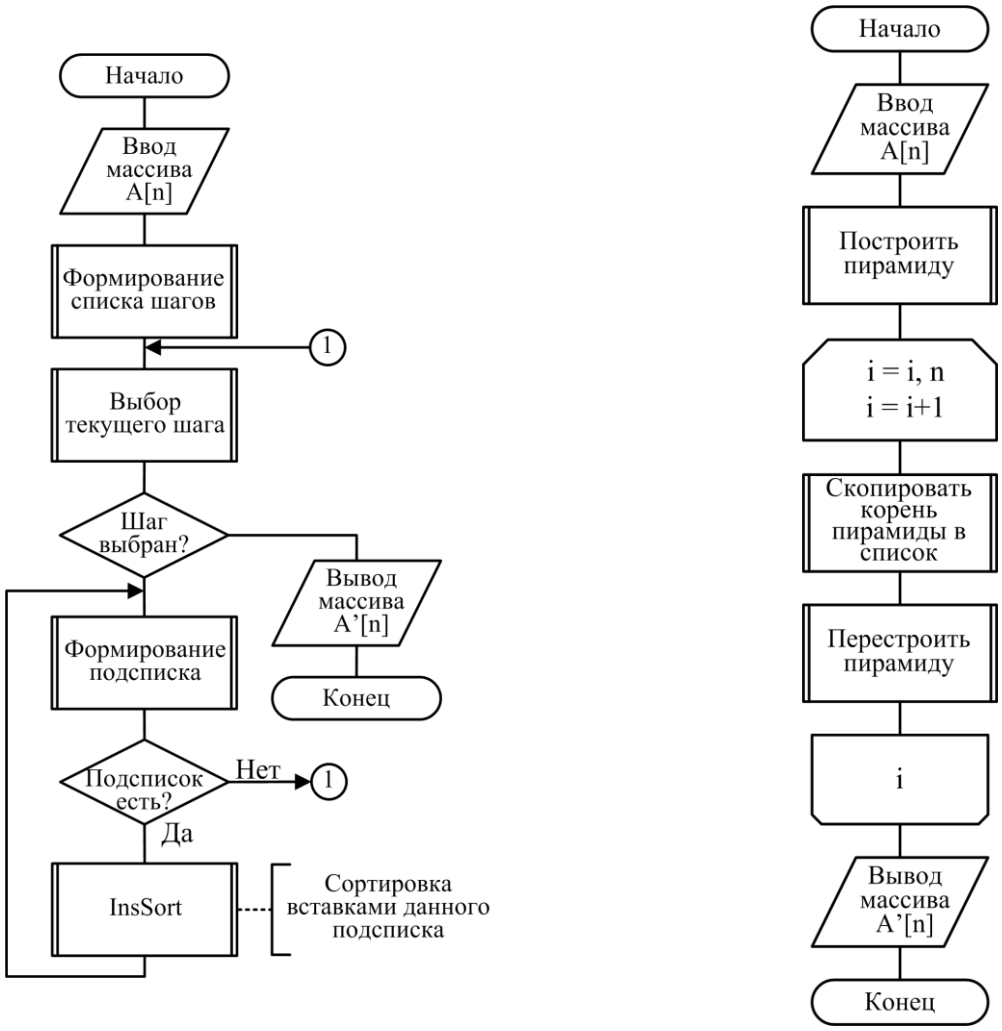

Рис. 5. Схема алгоритма сортировки Шелла (А) и алгоритма пирамидальной сортировки (Б)

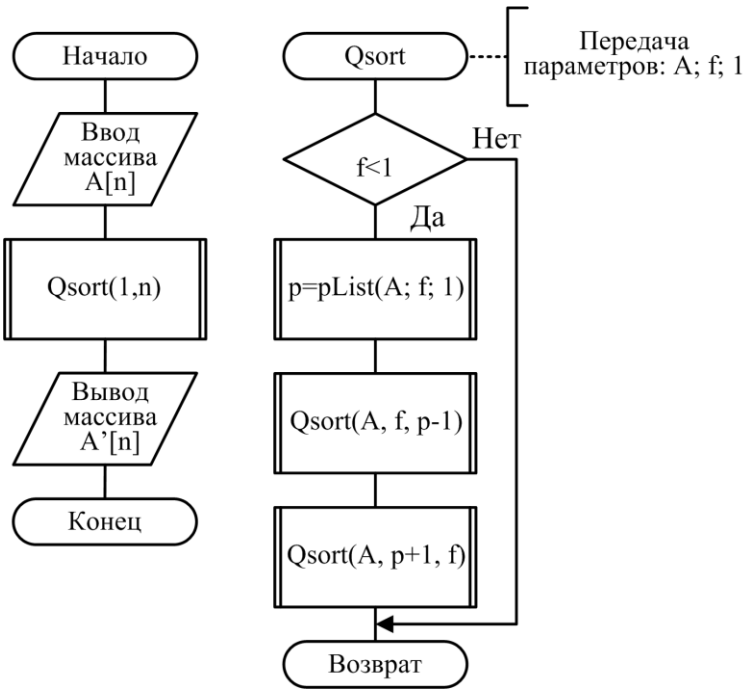

Рис. 6. Схема алгоритма быстрой сортировки

Приведём словесные описания СА (рис. 5, (А), 5, (Б), 6) и списки МО для эффективных сортировок.

Сортировка Шелла

Словесное описание

0. Ввод исходного массива.

1. Формирование списка шагов.

2. Главный цикл перебора шагов из списка.

А) для данного шага h в цикле формируются h подсписков;

Б) перебор подсписков. Для данного подсписка проводится сортировка вставками;

В) производится сборка подсписков в общий список (массив).

3. Проверяется исчерпание списка. В случае положительного ответа алгоритм заканчивает работу и выводится результат. Иначе – переход на п. 2, где из списка выбирается следующий шаг.

Список МО

1) МО «Ввод исходных данных» и МО «Вывод результатов»;

2) МО «Присваивание с вычислением» для вычисления шага между элементами подсписков;

3) МО «Формирование подсписков», которую целесообразно разделить на МО «Разделение подсписков» и МО «Объединение подсписков»;

4) МО «Поиск подходящего места» (цикл по условию 2-го рода), причём тело цикла – это МО «Сравнить – сдвинуть» (МО из сортировки вставками);

5) МО «Вставить элемент на подходящее место» (МО из сортировки вставками).

Быстрая сортировка

Словесное описание

0. Ввод исходного массива.

1. Выбор опорного элемента из массива.

2. Операция расщепления массива: преобразование массива таким образом, чтобы все элементы со значением меньшим или равным опорному элементу, оказались слева от него, а все элементы, превышающие по значению опорный – справа от него.

3. Рекурсивно упорядочиваем подмассивы, лежащие слева и справа от опорного элемента.

4. Базой рекурсии являются пустой набор или набор, состоящий из одного элемента, которые возвращаются в исходном виде. Все такие отрезки уже упорядочены в процессе разделения.

Примечание. Поскольку в каждой итерации (на каждом следующем уровне рекурсии) длина обрабатываемого отрезка массива уменьшается, по меньшей мере, на единицу, терминальная ветвь рекурсии будет достигнута обязательно, и обработка гарантированно завершится.

Список МО

1) МО «Ввод исходных данных» и МО «Вывод результатов»;

2) МО «Присваивание» для определения индекса и значения осевого элемента;

3) МО «Присваивание с вычислением» для определения изменения индекса осевого элемента;

4) МО «Сравнение» для сравнения осевого элемента с остальными элементами списка, а также левой и правой границ сортируемого списка;

5) МО «Обмен» для обмена местами двух элементов;

6) МО «Перемещение границы» (счётный цикл; граничным является осевой элемент);

7) МО "Разделение списка" (массива).

Пирамидальная сортировка

Словесное описание

0. Ввод исходного массива.

1. Построение пирамиды – бинарного дерева, для которого значение каждого корня поддерева больше значений его обоих потомков

2. Главный цикл перебора шагов из списка.

А) для данного шага i в цикле выполняется присоединение корня сформированной пирамиды к отсортированной последовательности;

Б) перестройка пирамиды, данные для нее берутся из оставшейся части неотсортированной последовательности;

3. Проверяется исчерпание списка. В случае положительного ответа алгоритм заканчивает работу и выводится результат. Иначе – переход на п. 2, где из списка выбирается следующий шаг.

Список МО

1) МО «Ввод исходных данных» и МО «Вывод результатов»;

2) МО «Присваивание с вычислением» для определения индекса середины массива, требуется для построения полной (или близкой к полной) пирамиды;

3) МО «Поиск максимального элемента» при анализе соответствия листьев поддеревьев их корням;

4) МО «Обмен элементов» листа поддерева и его корня;

5) МО «Перемещение границы» (счетный цикл) для определения места вставки элемента из корня пирамиды в отсортированную подпоследовательность;

6) МО «Присваивание» – непосредственно для самой вставки.

Систематизация МО

В работе выполнена сборка списков МО для каждой сортировки и проведен анализ МО на предмет их объединения в обобщенный список.

Общие МО: ввода, вывода, присваивания, присваивания с вычислением, перемещения границы.

Частные МО:

МО «Сравнить – выполнить» с вариантами

- сравнить элементы – обменять (переставить) элементы;

- сравнить особый элемент с текущим элементом – сдвинуть текущий элемент вправо (влево);

- сравнить минимальный элемент с текущим элементом – присвоить минимальному элементу значение текущего элемента.

МО «Поиск особого элемента» с вариантами:

- минимальный;

- максимальный;

- опорный (осевой) – выбор по правилу.

МО «Обмен элементов» с вариантами:

- пара рядом стоящих элементов;

- граничный и текущий элементы;

- граничный и особый элементы;

- элементы в листе поддерева и в его корне.

Обобщённые МО:

- **МО «Поиск подходящего места»** – это МО «Сравнить – сдвинуть» с конкретизацией;

- **МО «Частичное упорядочение массива»** – это попарное сравнение и обмен в счётном цикле на текущем проходе массива;

- **МО «Разделение списка (подсписка)»**;

- **МО «Объединение подсписков»**.

Заключение

В работе рассмотрены с единой позиции модели простых сортировок, в которых базовым является понятие «граница». Вычислительные процессы (ВП) для простых и избранных сложных (эффективных) сортировок представлены в виде схем алгоритмов. Из ВП этих сортировок выделены МО, которые собраны в соответствующие списки. Эти списки систематизированы путем объединения в общий список с последующей его минимизацией. Специфической для рассмотренных сортировок, как и следовало ожидать, является МО «Сравнить – переставить». Кроме того, для сложных сортировок специфической является МО "Формирование подсписков".

Приведенные модели и СА простых сортировок представляют самостоятельный интерес.

Некоторые МО могут быть использованы и в других ВП, не только в процессах сортировок.

Дальнейшего анализа требуют вопросы организации ввода-вывода, инициализации, рекурсии и др.

1. *Кнут Д.* Искусство программирования. Т.3 – Сортировка и поиск. – М.: Изд. Дом «Вильямс», 2000. – 832 с.
2. *Любченко В.С.* К проблеме создания модели параллельных вычислений // Труды Третьей международной конференции «Параллельные вычисления и задачи управления – PACO'2006». – М.: Институт проблем управления им. В.А. Трапезникова РАН, 2006. – С. 1359–1374.

3. *Котов В.Е.* Сети Петри. – М.: Наука,1984. – 160 с.
4. *Паулин О.Н., Поляков Ю.С., Шульгин В.А.* Представление вычислительных процессов сетями Петри // Научные труды SWorld. – Вып. 4 (41). Том 2. – Иваново: Научный мир, 2015. – С. 64–70.
5. *Паулин О.Н., Комлевая Н.О., Марулин С.Ю.* Об управлении вычислительными процессами // Труды 17-й международной научно-практической конференции СИЭТ-2016. – Одесса: «ART-V», 2016.
6. *Макконнелл Дж.* Анализ алгоритмов. Активный обучающий подход. 3-е доп. издание. М.: Техносфера, 2009. – 416 с.

References

1. Knuth, D. (2000) The Art of Computer Programming, 3: Sorting and Searching, Addison-Wesley Professional.
2. Liubchenko, V. (2006) About of the problem creating a model of parallel computing. Proceedings of the Third International Conference "Parallel Computations and Control Problems - PACO'2006». p. 1359-1374.
3. Kotov, V. (1984) Petri Nets. Science.
4. Paulin, O. (2015) Presentation of Petri nets computational processes. SWorld. V4(41). T 2. p. 64–70.
5. Paulin, O. & Komlevaya, N. & Marulin, S. (2015) About computational processes control. Modern information and electronic technologies 2016.
6. McConnell, J. (2007) Analysis of Algorithms: An Active Learning Approach, 2nd edn. Jones and Bartlett Publishers Inc.

Об авторах:

Паулин Олег Николаевич,
профессор, доктор технических наук.
Количество научных публикаций в украинских изданиях – 130.
Количество научных публикаций в иностранных изданиях – 2.
Индекс Гирша – 3.
http://orcid.org/0000-0002-2210-8317,

Комлева Наталия Олеговна,
доцент, кандидат технических наук.
Количество научных публикаций в украинских изданиях – 40.
Количество научных публикаций в иностранных изданиях – 2.
Индекс Гирша – 2.
http://orcid.org/0000-0002-2430-0134,

Марулин Станислав Юрьевич,
старший преподаватель, кандидат технических наук.
Количество научных публикаций в украинских изданиях – 26.
Количество научных публикаций в иностранных изданиях – 1.
Индекс Гирша – 1.
http://orcid.org/0000-0003-0755-0104.

Место работы авторов:

Одесский национальной политехнический университет,
65044, Украина, Одесса, пр. Шевченко, 1,

Институт компьютерных систем, кафедра Системного программного обеспечения.
Телефон: (048) 705 8328, 705 8566, 705 8698.
E-mail: paolenic@yandex.ua, nokoml@yandex.ua, stasfoot@mail.ru