ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ. ДЕРЕВЬЯ

Методические указания к выполнению лабораторных и самостоятельных работ

Самара 1994

Составители М.А.Кораблин, Е.В.Симонова

УДК 681.142.2

Динамические структуры данных. Деревья: Метод. указания к выполнению лабораторных и самостоятельных работ / Самар. госуд. аэрокосм. ун-т; Сост. М.А.Кораблин, Е.В.Симонова; Самара, 1994. 16 с.

Методические указания содержат краткие теоретические сведения по организации структур деревьев и варианты заданий для выполнения лабораторных и самостоятельных работ.

1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Деревья общего вида

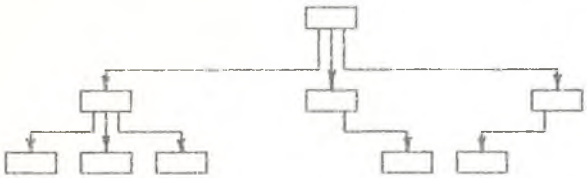
Нелинейные структуры данных выражают более сложные отношения порядка между объектами, чем отношения предшествования и следования. Наиболее важным видом нелинейных структур являются деревья.

Дерево - это конечное множество T, состоящее из одного или более узлов (вершин), для которых выполняются следующие условия:

a) имеется один специально выделенный узел, называемый корнем данного дерева;

b) остальные узлы (исключая корень) содержатся в m попарно не пересекающихся множествах T1, ..., Tm, каждое из которых в свою очередь является деревом. Деревья T1, ..., Tm называются поддеревьями данного корня.

Структура дерева общего вида представлена на рис. 1.



Р и с. 1. Дерево общего вида

Число поддеревьев данного узла называется степенью этого узла. Узел с нулевой степенью называется концевым узлом или листом. Уровень узла - выраженная в числе дуг длина пути, ведущего из данного узла в корень дерева. Считается, что корень дерева находится на уровне 0. Если некоторый узел 1 находится на уровне i, то находящийся непосредственно ниже его на уровне i+1 узел 2 называется

непосредственным потомком узла 1, а узел 1 в этом случае называется **предком** узла 2. Максимальный уровень какого-либо из узлов дерева - **глубина (или высота) дерева**. На множестве узлов дерева могут быть определены такие отношения порядка как "ПРЕДОК", "ПОТОМОК", "БРАТ" и т.п.

Основные свойства деревьев общего вида:

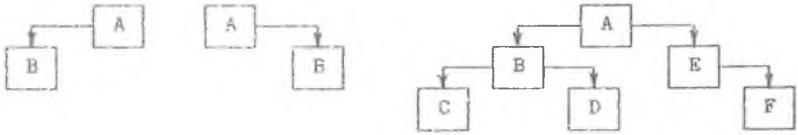
1. Корень не имеет предков;
2. Каждый узел, кроме корня, имеет только одного предка;
3. Каждый узел связан с корнем единственным путем, т.е. в деревьях отсутствуют замкнутые контуры.

Если в пункте b) определения дерева имеет значение относительный порядок поддеревьев T1,...,Tm, дерево является **упорядоченным**. Поэтому два упорядоченных дерева на рис.2 - это разные, отличные друг от друга объекты.

Р и с. 2. Два различных дерева

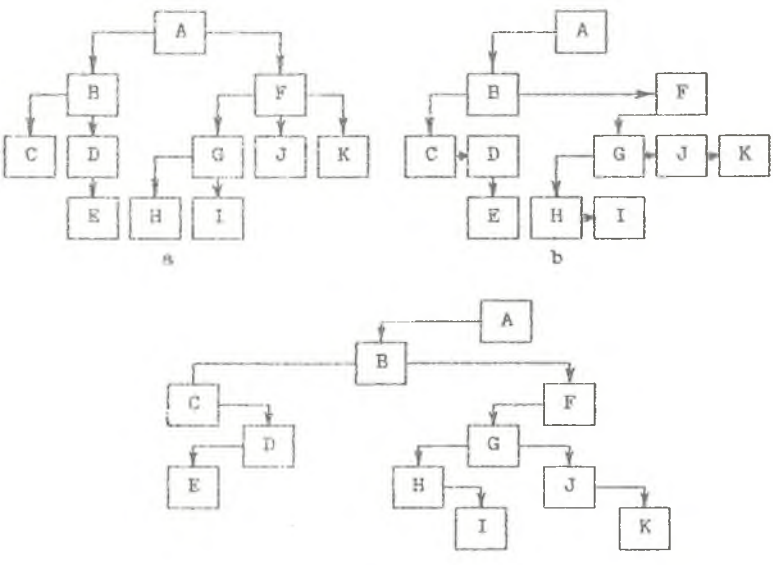
1.2. Бинарные деревья

Особенно важное практическое значение имеют упорядоченные деревья второй степени. Их называют двоичными (бинарными) деревьями. **Бинарное дерево** - это конечное множество узлов, которое или пусто, или состоит из корня и двух непересекающихся бинарных деревьев, называемых левым и правым поддеревьями данного корня. Примеры бинарных деревьев приведены на рис. 3.



Р и с. 3. Примеры бинарных деревьев

Существуют различные алгоритмы преобразования дерева произвольной степени к виду бинарного. Один из этих алгоритмов формулируется следующим образом: у каждого узла дерева общего вида необходимо сохранить самую левую связь, а узлы - потомки одного и того же узла соединить правой связью. На рис.4а представлено исходное дерево общего вида, на рис.4с - эквивалентное бинарное.

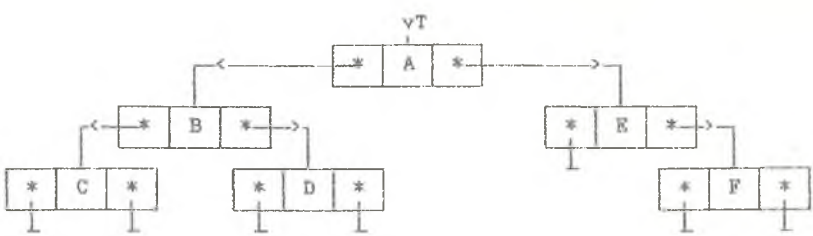


Р и с. 4. Представление произвольного дерева в виде бинарного

Элемент хранения узла бинарного дерева состоит из одного или нескольких информационных полей и двух полей связи, указывающих соответственно на левое и правое поддеревья данного узла.

```
TYPE
 PTree = POINTER TO Tree;        (* Указатель на узел дерева *)
 Tree = RECORD                      (* Описание узла дерева *)
  Info: CHAR;                        (* Информационное поле *)
  LLink, RLink: PTree               (* Ссылки на поддеревья *)
END;
VAR T: PTree;                   (* Указатель на корень дерева *)
```

Дерево задается указателем на его корень. Если дерево пусто (не существует), указатель на его корень равен NIL. Связанное представление бинарного дерева иллюстрирует рис. 5.



Р и с. 5. Связанное представление бинарного дерева

Наиболее типичная операция, выполняемая над бинарными деревьями - обход дерева. Обход - это процедура, при выполнении которой каждая вершина обрабатывается ровно один раз некоторым единым образом. Полный обход дерева дает линейную расстановку узлов, что облегчает выполнение многих алгоритмов.

Наиболее распространены три вида обходов: нисходящий (прямой) метод обхода, восходящий и смешанный.

Нисходящий обход выполняется по формуле "корень-левый-правый" (К Л П):

1. Обработать корневой узел.

2. В нисходящем порядке обойти левое поддерево.

3. В нисходящем порядке обойти правое поддерево.

Трасса нисходящего обхода дерева рис.5: A B C D E F.

Смешанный (обратный) метод обхода (Л К П):

1. В смешачном порядке обойти левое поддерево.

2. Обработать корневой узел.

3. В смешанном порядке обойти правое поддерево.

Трасса смешанного обхода дерева рис.5: C B D A E F.

Восходящий (концевой) метод обхода (Л П К):

1. В восходящем порядке обойти левое поддерево.

2. В восходящем порядке обойти правое поддерево.

3. Обработать корневой узел.

Трасса восходящего обхода дерева рис.5: C D B F E A.

Сбалансированным называется бинарное дерево, у которого число узлов в его левых и правых поддеревьях отличается не более чем на 1. Сбалансированное дерево, состоящее из n узлов, имеет минимальную высоту. Высоту сбалансированного дерева можно определить по формуле: $H = \lfloor \log_2 n \rfloor$, где $\lfloor \rfloor$ означает нижнюю границу величины.

Минимальная высота при заданном числе узлов достигается, если на всех уровнях, кроме последнего, размещается максимально возможное число узлов. Это достигается за счет равномерного размещения

узлов поровну слева и справа от каждого узла.

Правило такого размещения для n узлов (*правило1*):

1. Взять один узел в качестве корня.
2. Построить по *правилу1* левое поддерево с числом узлов
   nl = n DIV 2.
3. Построить по *правилу1* правое поддерево с числом узлов
   nr = n-nl-1.

Двоичные деревья часто употребляются для представления множества данных, среди которых идет поиск элементов по уникальному значению некоторого атрибута, называемого ключом. При этом на множестве элементов определено особое отношение порядка - *дихотомия*, заключающееся в поэтапном разбиении множества информационных полей элементов на два непересекающихся подмножества. Дихотомическим деревом (деревом поиска) называется бинарное дерево, организованное так, что для каждого узла с ключом T1 справедливо утверждение, что все ключи его левого поддерева меньше ключа T1, а все ключи его правого поддерева больше T1.

```
TYPE
 PTree = POINTER TO Tree;       (* Указатель на узел дерева *)
 Tree = RECORD                (* Описание узла дерева поиска *)
  Key:  CARDINAL;                              (* Поле ключа *)
  Info: CHAR;                          (* Информационное поле *)
  LLink, RLink: PTree                 (* Ссылки на поддеревья *)
 END;
```

Поиск в таком дереве элемента с заданным ключом тривиален: начав с корня, перейти к левому или правому поддереву на основании

лишь одного сравнения с ключом текущего узла. Из n элементов можно организовать двоичное дерево высотой не более log(n). Поэтому, если дерево является сбалансированным, поиск среди n элементов выполняется максимум за log(n) сравнений. Поиск по дереву с включением нового узла выполняется следующим образом: если поиск узла с заданным значением ключа приводит в тупик (т.е. к пустому поддереву, обозначенному ссылкой со значением NIL), то новый узел необходимо включить в дерево на место пустого поддерева.

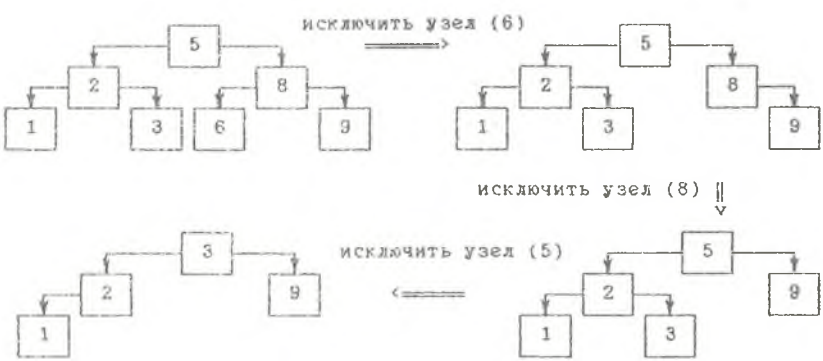
При удалении из бинарного дерева поиска узла с ключом X различают 3 случая:

1. Узла с ключом X в дереве нет.

2. Узел с ключом X имеет не более одного потомка. В этом случае удаляемый элемент заменяется этим потомком.

3. Узел с ключом X имеет двух потомков. В этом случае удаляемый элемент заменяется либо на *самый правый элемент* его левого поддерева, либо на *самый левый элемент* его правого поддерева.

Пример исключения узлов из бинарного дерева приведен на рис.6.



Р и с. 6. Исключение узлов из бинарного дерева

1.3. Рекурсия

Рекурсия - есть метод определения множества или процесса в терминах самого себя. Например, приведенное выше определение дерева, алгоритмы обхода или правило1 рекурсивны. Существуют рекурсивные структуры данных и рекурсивные процедуры. Примером рекурсивных структур являются деревья, списки. Процедура, содержащая обращение к самой себе, называется рекурсивной процедурой.

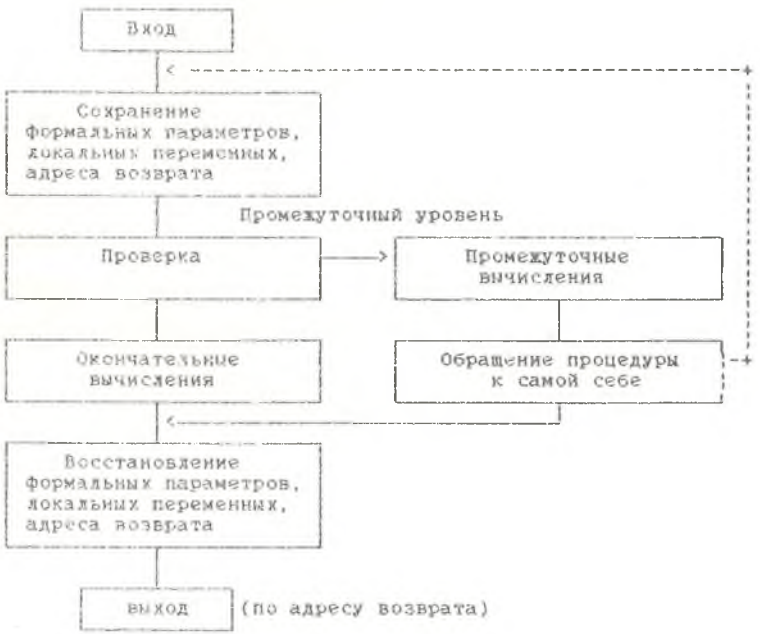
Применять рекурсию целесообразно для обработки данных, имеющих рекурсивную структуру. В этом случае рекурсивный процесс оказывается эффективнее, чем итеративный. Любой итеративный алгоритм имеет рекурсивный эквивалент (обратное неверно).

Итеративный процесс можно проиллюстрировать с помощью схемы, приведенной на рис.7. Этот процесс состоит из четырех частей: инициализации, принятия решения (о продолжении вычислений), вычисления и модификации.

Р и с. 7. Схема итеративного процесса

В отличие от итеративной процедуры существуют особые проблемы, связанные с использованием рекурсивной процедуры. К рекурсивной процедуре можно обращаться как из нее самой, так и извне, поэтому

для обеспечения правильного функционирования она должна сохранять адреса возврата в таком порядке, чтобы возврат из нее производился в надлежащее место вслед за оператором вызова. Процедура должна также сохранять формальные параметры, локальные переменные и т.д. при входе и восстанавливать все эти параметры и переменные при выходе из процедуры. Общая схема рекурсивной процедуры представлена на рис.8.



Р и с. 8. Схема рекурсивной процедуры

Тело процедуры содержит блоки промежуточных и окончательных вычислений. Нередко блок промежуточных вычислений объединяется с блоком обращения к процедуре (например, при вычислении факториа-

да). В блоке окончательных вычислений производится явное определение параметров-переменных процедуры для конкретных значений входных параметров. В блоке проверки определяется, являются ли значения входных параметров такими, для которых возможно вычисление значений выходных параметров.

С каждым обращением к рекурсивной процедуре ассоциируется **номер уровня**. Вход в процедуру при первоначальном обращении из основной программы характеризуется номером уровня 1, при этом предполагается, что основная программа имеет номер уровня 0. Каждый последующий вход в процедуру имеет номер уровня на 1 больше, чем номер уровня процедуры, из которой производится это обращение. Другой характеристикой рекурсивной процедуры является **глубина рекурсии**, определяемая числом рекурсивных обращений к процедуре в процессе вычисления при заданных аргументах. В общем случае эта величина неочевидна, исключение составляют простые рекурсивные функции: например, для вычисления функции FACTORIAL(N) глубина рекурсии равна N.

Перед выходом из рекурсивной процедуры восстанавливаются параметры, локальные переменные и адрес возврата, которые были сохранены самыми последними, поэтому стек является наиболее подходящей структурой данных, которую следует использовать для сохранения текущего состояния рекурсивной процедуры. При каждом обращении к процедуре (или на каждом уровне рекурсии) происходит "проталкивание" в стек для сохранения необходимых величин; при выходе из данного уровня осуществляется "выталкивание" из стека для восстановления сохраняемых величин предыдущего уровня.

Рекурсивный механизм проиллюстрируем на примере процедуры нисходящего обхода бинарного дерева, выводящей на печать трассу обхо-

да - последовательность информационных полей вершин дерева.

```
            TYPE
             PTree = POINTER TO Tree;
             Tree = RECORD
              Info: CHAR;
              LLink, RLink: PTree
             END;
             PROCEDURE Nis( T: PTree );
             BEGIN
              IF T # NIL THEN
               WrChar( T^.Info );
               Nis( T^.LLink );
(* 1 *)       Nis( T^.RLink );
(* 2 *)     END
(* 3 *)   END Nis;
```

Для иллюстрации работы этой процедуры и построения трассы состояний стека будем использовать следующие обозначения:

- ^A - адрес узла дерева, в информационном поле которого хранится символ "A";

- (* 1 *) и (* 2 *) - адреса возврата из уровня рекурсивной процедуры, номер которого на 1 больше текущего;

- (* 3 *) - адрес возврата из рекурсивной процедуры текущего уровня.

Трасса нисходящего обхода дерева, приведенного на рис.5, содержится в следующей таблице.

Трасса нисходящего обхода бинарного дерева

N входа в процедуру	Значение фактического параметра	Стек		Выходная трасса	Выход из уровня
		Исходное состояние	Конечное состояние		
1	^A	(- , -)	(^A , *1*)	A	
2	^B	(^A , *1*)	(^B , *1*) (^A , *1*)	B	
3	^C	(^B , *1*) (^A , *1*)	(^C , *1*) (^B , *1*) (^A , *1*)	C	
4	NIL	(^C , *1*) (^B , *1*) (^A , *1*)	(^B , *1*) (^A , *1*)		(*3*)
	^C	(^B , *1*) (^A , *1*)	(^C , *2*) (^B , *1*) (^A , *1*)		
5	NIL	(^C , *2*) (^B , *1*) (^A , *1*)	(^B , *1*) (^A , *1*)		(*3*)
	^C	(^B , *1*) (^A , *1*)	(^A , *1*)		(*3*)
	^B	(^A , *1*)	(^B , *2*) (^A , *1*)		
6	^D	(^B , *2*) (^A , *1*)	(^D , *1*) (^B , *2*) (^A , *1*)	D	
7	NIL	(^D , *1*) (^B , *2*) (^A , *1*)	(^B , *2*) (^A , *1*)		(*3*)
	^D	(^B , *2*) (^A , *1*)	(^D , *2*) (^B , *2*) (^A , *1*)		
8	NIL	(^D , *2*) (^B , *2*) (^A , *1*)	(^B , *2*) (^A , *1*)		(*3*)
	^D	(^B , *2*) (^A , *1*)	(^A , *1*)		(*3*)
	^B	(^A , *1*)	(- , -)		(*3*)
	^A	(- , -)	(^A , *2*)		

N входа в процедуру	Значение фактического параметра	Стек		Выходная трасса	Выход из уровня
		Исходное состояние	Конечное состояние		
9	^E	(^A , *2*)	(^E , *1*) (^A , *2*)	E	
10	NIL	(^E , *1*) (^A , *2*)	(^A , *2*)		(*3*)
	^E	(^A , *2*)	(^E , *2*) (^A , *2*)		
11	^F	(^E , *2*) (^A , *2*)	(^F , *1*) (^E , *2*) (^A , *2*)	F	
12	NIL	(^F , *1*) (^E , *2*) (^A , *2*)	(^E , *2*) (^A , *2*)		(*3*)
	^F	(^E , *2*) (^A , *2*)	(^F , *2*) (^E , *2*) (^A , *2*)		
13	NIL	(^F , *2*) (^E , *2*) (^A , *2*)	(^E , *2*) (^A , *2*)		(*3*)
	^F	(^E , *2*) (^A , *2*)	(^A , *2*)		(*3*)
	^E	(^A , *2*)	(- , -)		(*3*)
	^A	(- , -)	(- , -)		(*3*)

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Лабораторная работа выполняется в четыре этапа.

1.Ознакомительный, на котором студент изучает методы представления и обработки динамических структур данных.

2. Подготовительный, на котором составляются алгоритм и программа работы с динамическими структурами данных.

3. Лабораторный, на котором производится отладка программ.

4. Составление отчета.

Отчет должен содержать:

- название лабораторной работы и текст варианта задания;

- листинг программы.

Для сдачи отчета необходимо продемонстрировать работу программы на ЭВМ. Информация, необходимая для выполнения работы и опущенная в тексте задания, должна быть определена самостоятельно.

Требования к программе:

- корректность исходных данных, вводимых с клавиатуры, должна контролироваться Вашей программой;

- работа программы должна иллюстрироваться средствами графического и текстового вывода;

- взаимодействие пользователя с Вашей программой должно осуществляться через систему "меню", обеспечивающую многократное переопределение исходных данных и завершение работы программы.

3. ВАРИАНТЫ ЗАДАНИЙ

1. Создать сбалансированное дерево. Найти среднее арифметическое значений информационных полей узлов дерева.

2. Создать сбалансированное дерево. Подсчитать количество узлов дерева с положительными и отрицательными значениями информационных полей.

3. Создать сбалансированное дерево. Подсчитать количество узлов дерева с заданным значением информационных полей.

4. Создать дерево поиска. Подсчитать сумму значений информационных полей узлов дерева.

5. Создать дерево поиска. Подсчитать количество листьев в дере-

ве.

6. Создать сбалансированное дерево. Подсчитать количество узлов дерева, не являющихся листьями.

7. Создать сбалансированное дерево. Подсчитать количество узлов дерева с заданным значением информационного поля.

8. Создать дерево поиска. Заменить все отрицательные значения информационных полей узлов дерева на их абсолютные величины.

9. Создать сбалансированное дерево. Найти максимальное значение среди всех информационных полей узлов дерева.

10. Создать дерево поиска. Построить копию этого дерева.

11. Создать дерево поиска. Скопировать в линейный список узлы дерева, выбранные по заданным значениям ключей.

12. Создать дерево поиска. Скопировать в новое дерево поиска узлы дерева, выбранные по заданным значениям ключей.

13. Создать сбалансированное дерево. Скопировать в упорядоченный линейный список узлы дерева, значения информационных полей которых лежат в диапазоне от N до K.

14. Создать сбалансированное дерево. Получить два линейных упорядоченных списка: в 1-й список скопировать узлы дерева, у которых значение информационного поля больше N; во 2-й список скопировать все остальные узлы дерева.

15. Создать два сбалансированных дерева. Определить эквивалентность двух деревьев.

16. Создать дерево поиска. Определить глубину дерева.

17. Создать дерево поиска. Написать процедуру исключения произвольного узла из дерева.

18. Задано произвольное дерево, не являющееся бинарным. Напишите процедуру преобразования этого дерева в бинарное.

19. Заданы два двухуровневых дерева, первое определяет потомков мужчины A, второе - потомков женщины B. Напишите процедуру, определяющую общих потомков - детей A и B.

20. Задана символьная строка, определяющая структуру оператора присваивания: A := <Арифметическое выражение>. Арифметическое выражение допускает использование скобочной записи. Постройте бинарное дерево, структурно эквивалентное арифметическому выражению, и напишите процедуру обхода, реализующую вычисление этого выражения.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение дерева общего вида.
2. Перечислите свойства деревьев общего вида.
3. Дайте определение бинарного дерева.
4. Какие алгоритмы обхода бинарных деревьев вы знаете?
5. Дайте определение сбалансированного дерева. В чем отличительные особенности сбалансированных деревьев? Сформулируйте алгоритм построения сбалансированного дерева.
6. Дайте определение дихотомического дерева. Приведите примеры применения дихотомических деревьев.
7. Сформулируйте алгоритм создания дихотомического дерева.
8. Сформулируйте алгоритм исключения произвольного узла из дихотомического дерева.
9. Дайте определение механизма рекурсии. Чем отличается организация итеративного вычислительного процесса от рекурсивного?
10. Что такое глубина рекурсии?

Динамические структуры данных. Деревья

Составители: Кораблин Михаил Александрович

Симонова Елена Витальевна

Редактор Л.Я.Чегодаева

Техн. редактор Г.А.Усачева

Подписано в печать 28.06.94. Формат 60 x 84¹/16.

Бумага офсетная. Печать офсетная.

Усл. печ. л. 0,93. Усл.кр.-отт. 1,05. Уч.-изд.л. 1,0.

Тираж 150 экз. Заказ 257. . Арт.С - 106 / 94.

Самарский государственный аэрокосмический

университет имени академика С.П.Королева.

443086 Самара Московское шоссе, 34.

ИПО Самарского государственного аэрокосмического

университета. 443001 Самара, ул.Ульяновская, 18.