МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Брянский государственный технический университет

Утверждаю
Ректор университета

_____О.Н. Федонин

«____»_____2014 г.

СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

ОЧЕРЕДИ И СТЕКИ

Методические указания
к выполнению лабораторной работы №4
для студентов очной, очно-заочной и заочной форм обучения
по направлениям подготовки
09.03.01 «Информатика и вычислительная техника»,
02.03.03 «Математическое обеспечение и администрирование информационных систем»,
09.03.04 «Программная инженерия»

Брянск 2017

УДК 006.91

Структуры и алгоритмы обработки данных. Очереди и стеки [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы №4 для студентов очной, очно-заочной и заочной форм обучения по направлениям подготовки 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем», 09.03.04 «Программная инженерия». – Брянск: БГТУ, 2014. – 25 с.

Разработал:
канд. техн. наук, проф.
В.К. Гулаков

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол №1 от 13.09.17)

1. ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Целью лабораторной работы является приобретение навыков использования стеков и очередей.

Продолжительность лабораторной работы − 4 часа.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В полустатических структурах данных изменяется размер структуры, а связи между элементами структуры не изменяются. К таким структурам данных относятся: стек, очередь, дек.

Стек

Стек – последовательность переменной длины, включение и исключение элементов из которой выполняется только с одной стороны последовательности по принципу «последним пришел, первым ушел».

Над стеком выполняются следующие операции:

– вершина стека адресуется с помощью специального указателя;

– включение элементов осуществляется с вершины стека. При этом указатель сначала перемещается вверх на длину слота, или ячейки, а затем по значению указателя в стек помещается новый элемент;

– исключение элементов осуществляется с вершины стека. Сначала прочитывается информация об исключаемом элементе по значению указателя, а затем указатель смещается «вниз» на один слот;

– очистка стека;

– проверка объема стека.

Существуют следующие способы реализация стека:

- статическая реализация стека на основе массива;
- динамическая реализация стека на основе односвязного списка;
- реализация стека на основе кольцевого списка.

Статическая реализация стека на основе массива

На логическом уровне стек представляется в виде массива (рис.1).

Рис.1. Представление стека на логическом уровне

Для хранения стека отводится сплошная область памяти. Граничные адреса этой области памяти являются параметрами физической структуры стека. Длина стека изменяется при его использовании, однако изменения не должны превышать пределов выделенной памяти. Вот поэтому стек – полустатическая структура данных.

На физическом уровне стек дополняется обычно дескриптором, в котором кроме указателя содержатся, имя стека, границы физической структуры в памяти, описание элемента.

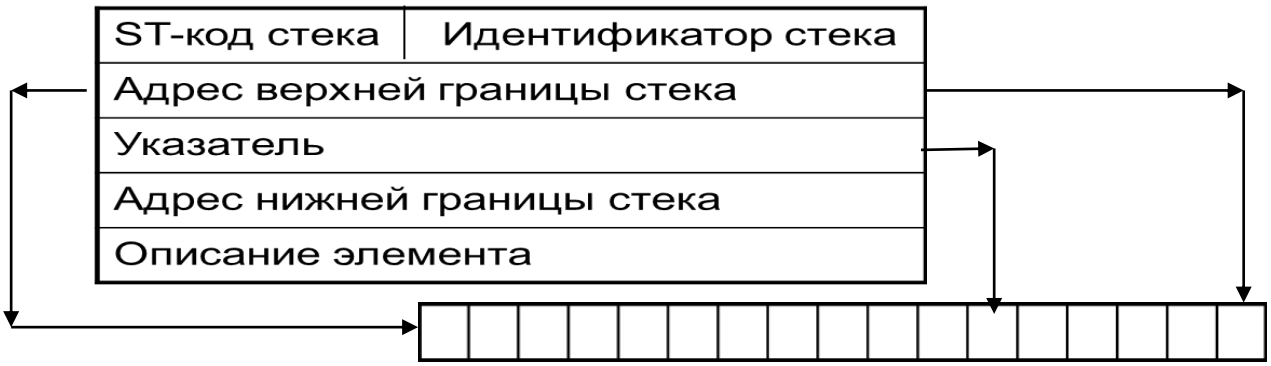

Рис. 2. Представление стека на физическом уровне

Пример 1. Применение стека для проверки правильности расстановки скобок в выражении:

$\{x+(y-[a+b])\cdot[c-(d+e)]\}/(h-(j-(k-[i-n])))$

Убедимся в следующем:

1) что число левых и правых скобок одинаково;
2) каждой закрывающей скобке предшествует открывающаяся;

3) в выражении имеется три типа скобок: квадратные [], фигурные {}, и круглые ().

													[			
		[				(						(	(	(		
	(	(	(		[	[	[				(	(	(	(	(	
{	{	{	{	{	{	{	{	{		(	(	(	(	(	(	(

Рис. 3. Стек для проверки правильности расстановки скобок

Тип последней открывающей скобки должен совпадать с типом закрываемой. Это имитируется стеком, в котором последний размещаемый элемент удаляется первым. Каждый элемент стека представляет скобку, которая открыта, но не закрыта.

Стек применим в любой ситуации, когда применим принцип «последний размещенный извлекается первым».

Прежде чем рассматривать другие задачи применения стеков, рассмотрим несколько понятий.

Существуют три формы записи выражений: инфиксная, префиксная и постфиксная. Префиксы «пре», «пост» и «ин» относятся к относительной позиции оператора по отношению к обоим операндам.

Инфиксная	Префиксная	Постфиксная
A+B	+AB	AB+
A+B-C	-+ABC	AB+C-
(A+B)*(C-D)	*+AB-CD	AB+CD-*
((A+B)*C-(D-E))↑(F+G)	↑-*+ABC-DE+FG	AB+C*DE--FG+↑
A-B/(C*D↑E)	-A/B*C↑DE	ABCDE↑*/-

A+B – инфиксная запись, т.е. традиционная, включающая скобки,

+AB – префиксная запись,

AB+ – постфиксная запись,

Записи без скобок и операнды не имеют приоритета, т.е. вычисление выражений идет значительно проще, так как не требуется тратить ресурсы на определение приоритета операции.

Приоритетность операции учитывается при получении префиксной или постфиксной записи.

В дальнейшем будем рассматривать постфиксные выражения

Пример 2. Применение стека для вычисления выражения, записанного в постфиксной форме.

- Прочитываем и записываем в стек операнд.
- Когда достигаем операции, то относящиеся к ней операнды будут двумя верхними элементами стека.
- Затем можно извлечь эти два операнда, осуществить над ними указанную операцию и записать результат в стек.
- Этот результат в качестве операнда станет доступен для следующей операции.

Рассмотрим выражение $(6-(2+3))\cdot(3+8/2)^2+3$ в постфиксной нотации 6 2 3 + – 3 8 2 / + * 2 ↑ 3 +

Принцип «последнее открываемое выражение вычисляется первым» предполагает использование стека.

Символ выражения	Операнд 1	Операнд 2	Результат	Стек
6				6
2				6 ,2
3				6 ,2,3
+	2	3	5	6 ,5
-	6	5	1	1
3				1 ,3
8				1 ,3,8
2				1 ,3,8,2

/	8	2	4	1,3,4
+	3	4	7	1,7
*	1	7	7	7
2				7,2
↑	7	2	49	49
3				49,3
+	49	3	52	52

Организация стека в виде массива целесообразна, когда имеется один, достаточно большой непрерывный участок памяти.

Стеки применяют не только при создании и обработке структур данных, но и оказали влияние на выбор архитектуры ЭВМ, послужив основой для стековых машин. Наличие стека дает возможность расширять спектр «безадресных» команд, т.е. команд, не требующих явного задания адресов операндов, и тем самым увеличить скорость выполнения соответствующих операций.

Очередь

Очередь – такая последовательность переменной длины, включение элементов в которую происходит с одной стороны, а исключение с другой стороны последовательности.

Та сторона, с которой происходит добавление элементов, называется хвостом, или концом, очереди, другая головой. Для индикации хвоста и головы организуется два указателя.

Основными операциями над очередью являются:

- включение элемента;
- исключение элемента;
- проверка переполнения;
- организация кольцевой очереди.

Представление очереди. Схема простейшей очереди – конечная последовательность ячеек или слотов, из которых в данный момент времени занято только часть последовательных слотов (рис. 4).

Возможны:

- статическая реализация очереди на основе массива;
- динамическая реализация очереди на основе односвязного списка;
- реализация очереди на основе кольцевого списка.

Слоты имеют адреса *A*1 , *A*2, ..., *Amax*

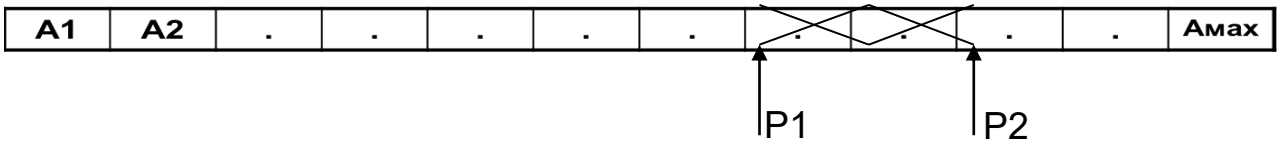

Рис. 4. Представление очереди

Каждый элемент очереди представляет собой в общем случае запись с одинаковой организацией для всех элементов. Идентификаторы *P*1 и *P*2 – указатели, причем *P*1 – указатель головного элемента очереди, а *P*2 адресует первый свободный слот, следующий за хвостовым элементом.

Кольцевая очередь. При включении элементов неизбежно наступит состояние переполнения. Этот недостаток можно устранить, если после достижения указателем слота с адресом *Amax* переводить этот указатель на слот с адресом *A*1, если он пуст. Организованная таким образом очередь называется кольцевой.

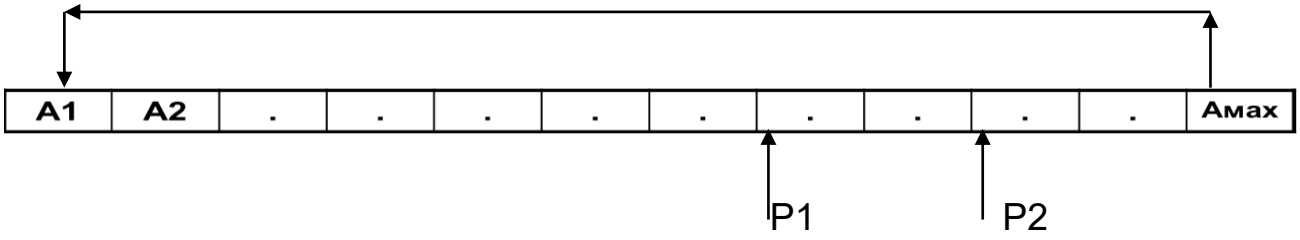

Рис. 5. Кольцевая очередь

Рис. 6. Дескриптор физической структуры очереди

Дек

Дек – последовательность, в которой как включение, так и исключение элементов может осуществляется с любого из двух концов последовательности. По своим возможностям дек объединяет возможности стека и очереди.

Дек с ограниченным входом может быть использован как простая очередь или как стек

Иногда дек называют двусторонней очередью.

Логическая и физическая структуры дека аналогичны логической и физической структуре кольцевой очереди. Однако, применительно к деку целесообразно говорить не о начале и конце, а о левом и правом конце

Возможны следующие реализации дека:

- статическая реализация дека на основе массива;
- динамическая реализация дека на основе двусвязного списка;
- реализация дека на основе кольцевого двусвязного списка.

Задачи, требующие структуры дека, встречаются в вычислительной технике и программировании значительно реже, чем задачи, реализуемые на структуре стека или очереди. Как правило, вся организация дека выполняется программистом без каких-либо специальных средств системной поддержки.

Примером дека может быть, например, некий терминал, в который вводятся команды, каждая из которых выполняется какое-то время. Если ввести следующую команду, не дождавшись, пока закончится выполнение предыдущей, то она встанет в очередь и начнет выполняться, как только освободится терминал. Это очередь. Если же дополнительно ввести операцию отмены последней введенной команды, то получается дек.

3. ТЕХНИКА БЕЗОПАСНОСТИ ПРИ ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

Меры безопасности при работе с электротехническими устройствами соответствуют мерам безопасности, принимаемым при эксплуатации установок с напряжением до 1000 В и разработанным в соответствии с «Правилами техники безопасности при эксплуатации электроустановок потребителей», утвержденными Главгосэнергонадзором 21 декабря 1984 г.

Студенту не разрешается приступать к выполнению лабораторной работы, если замечены какие-либо неисправности в лабораторном оборудовании.

Студент не должен прикасаться к токоведущим элементам электрооборудования, освещения и электропроводке, открывать двери электрошкафов и корпусов системных блоков, мониторов.

Студенту запрещается прикасаться к неизолированным или поврежденным проводам и электрическим устройствам, наступать на переносные электрические провода, лежащие на полу, самостоятельно ремонтировать электрооборудование и инструмент.

Обо всех замеченных неисправностях электрооборудования студент должен немедленно поставить в известность преподавателя или лаборанта.

При выполнении лабораторной работы необходимо соблюдать осторожность и помнить, что только человек, относящийся серьезно к своей безопасности, может быть застрахован от несчастного случая.

4. УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Каждую задачу решить в соответствии с изученными методами формирования, вывода и обработки данных очередей и стеков в языке C++. Обработку очередей или стеков выполнить на основе базовых алгоритмов: поиск, вставка элемента, удаление элемента, удаление всей динамической структуры. При объявлении списков выполните комментирование используемых полей. Задачи 2, 4 имеют исследовательский характер, поэтому при составлении отчета к ним подробно описать предлагаемый метод оценки максимального размера очереди или стека. Программу для решения каждой задачи разработать методом процедурной абстракции, оформив комментарии к коду.

Каждую задачу реализовать в соответствии с приведенными этапами:

- изучить словесную постановку задачи, выделив при этом все виды данных;
- сформулировать математическую постановку задачи;
- выбрать метод решения задачи, если это необходимо;
- разработать графическую схему алгоритма;
- записать разработанный алгоритм на языке C++;
- разработать контрольный тест к программе;
- отладить программу;
- составить отчет о лабораторной работе.

5. ЗАДАЧИ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Составить программу работы с циклической очередью, организованной с помощью массива. Обеспечить операции постановки в очередь, продвижения очереди, вставки в середину после элемента с заданным ключом и удаления из середины.

2. На узловой станции необходимо менять направления движения всех поездов. Для этого предназначен специальный тупик. Зашедший в тупик последний поезд выходит из него первым. Известны моменты прихода поездов и минимально необходимое время стоянки (одинаковое для всех поездов). Требуется:

а) составить расписание стоянки поездов на станции с учетом смен направления движения;

б) поменять между собой моменты прихода скорых и пассажирских поездов так, чтобы скорые поезда имели минимальное суммарное время простоя в тупике.

3. Задачи для выполнения на ЭВМ имеют различные приоритеты, задаваемые цифрами от 1 до 5 (5-высший приоритет). Для каждого приоритета образуется отдельная очередь. Приоритет задач может меняться. При повышении приоритета задание помещается в конец другой очереди, а при понижении - в начало. Составить программу, обеспечивающую:

а) выдачу общей очереди;

б) выдачу очереди заданного приоритета;

в) перестройку очередей при изменении приоритета.

4. Имеется некоторое изделие. Детали изделия могут быть составными и неразборными. Записи файла, описывающего изделие, содержат код детали и список кодов деталей, на которые разбирается данная деталь. Составить список кодов неразборных деталей.

5. Текст программы на ПАСКАЛЕ содержит примечания, выделенные фигурными скобками «{', '}» либо парами символов « (*' и '*)». Примечания могут быть вложенными друг в друга. Если примечание открыто знаком «{», то оно должно быть закрыто знаком «}». Аналогично примечание, начинающееся с символов «(*' должно заканчиваться символами '*)». Требуется:

а) проверить правильность вложенности примечаний;

б) переписать файл с исходным текстом так, чтобы отсутствовали вложенные примечания и в качестве ограничивающих символов остались только фигурные скобки.

6. Трассировка программы, не содержащей рекурсивных вызовов, распечатана в виде списка выполняемых процедур. Процедура попадает в список, если к ней произошло обращение из вызывающей процедуры либо возврат управления из вызванной ей процедуры. Структура программы такова, что каждая вызываемая

процедура вложена в вызывающую ее процедуру. Известен объем памяти, который требуется для загрузки каждой процедуры. При выходе из процедуры, занимаемая ей память освобождается. Определить размер основной памяти, которая требуется для выполнения программы.

7. Перед открытием двух железнодорожных касс сформировались 2 очереди пассажиров, причем некоторые из них находятся сразу в обеих очередях. Для каждого пассажира известны его места в очередях и необходимое время обслуживания. Если у какого-либо пассажира очереди подходят одновременно, то он обслуживается в первой кассе. Промоделировать работу касс, выдавая последовательно информацию об обслуживании пассажиров.

8. В некотором языке программирования операторы располагаются в разных строках. Строки программы нумеруются. Программист указал множество пар (m, n) , где m − номер начальной строки цикла, а n − номер конечной строки. Выяснить:

а) пропустит ли такой вариант транслятор?

б) правильно ли организовано вложение циклов?

9. В символьной строке записано выражение из букв и операций в постфиксной форме (знак операции после операндов). Проверить правильность записи и перевести выражение в инфиксную (обычную) форму со скобками с помощью стека.

Пример: выражение $(a+b)\cdot c-d\cdot e$ записывается в постфиксной форме как $ab+c\cdot de\cdot -$. Требуется выдать его в виде ($((a+b)\cdot c)-(d\cdot e)$).

10. Автомобильная стоянка вмещает n машин и имеет одну полосу с единственным въездом-выездом. Если владелец приходит за машиной, временно выезжают и потом возвращаются в том же порядке все машины, загораживающие проезд. Если стоянка заполнена, то прибывшая машина уезжает. Задана последовательность номеров машин с признаками прибытия или убытия. Составить протокол работы стоянки, сообщая обо всех событиях.

11. Два стека размещены в одном массиве и растут навстречу друг другу. Задана последовательность операций размещения и удаления элементов с указанием номера стека. Обеспечить выполнение данных операций и обработку аварийных ситуаций. Сообщить в конце:

а) какой минимальный резерв памяти оставался в массиве при выполнении операций;

б) какой интервал индексов массива не использовался для размещения элементов.

12. Трассировка программы, не содержащей рекурсивных вызовов, задана в виде списка выполняемых процедур. Процедура попадает в список, если к ней произошло обращение из вызывающей процедуры либо возврат управления из вызванной ей процедуры. Структура программы такова, что каждая вызываемая процедура вложена в вызывающую ее процедуру. Начало и окончание программы должно быть в головной процедуре. Выяснить, соответствует ли трассировка правильной работе программы.

13. Описать стек с целочисленным информационным полем. Заполнить его длинами строк, считанных из файла. Распечатайте на экране содержимое стека. Указать номер и длину последней самой короткой строки файла.

14. Разработать программу, с помощью которой можно определить наибольший допустимый размер очереди с вещественным информационным полем. Найдите этот размер (число элементов в очереди).

15. Описать очередь с вещественным информационным полем, и заполнить ее элементами с клавиатуры. Выполнить циклический сдвиг элементов в очереди так, чтобы в ее начале был расположен наибольший элемент.

16. Разработать программу, с помощью которой можно определить наибольший допустимый размер стека с вещественным информационным полем. Найти этот размер (число элементов в стеке). Сравните с наибольшим допустимым размером очереди с аналогичным информационным полем.

17. В некотором институте приобретаемые компьютеры выделяются различным факультетам поочередно. В пределах факультета имеются очереди из кафедр. Факультет, получивший компьютер, перемещается в конец очереди, а соответствующая кафедра исключается из факультетской очереди. Вновь организованные факультеты и кафедры занимают последние места в соответствующих очередях. Составить программу ведения очереди на компьютеры.

18. Данные о студенческих группах записаны в файле в виде

Информация о группе	Факультет	Курс

Организовать с помощью указателей размещение списка с данными о группах в основной памяти так, чтобы каждый элемент располагался один раз. Составить программу выдачи всех групп заданного факультета либо заданного курса без перебора всех элементов.

19. Выборы старосты в группе студентов из *M* человек организованы по следующим правилам. Задаются целые числа *N* и *K*. Студенты становятся по кругу в соответствии со своими номерами в журнале. От *N*-го студента отсчитывается *K*-й студент. Счет ведется циклически по возрастанию номеров. Этот студент выбывает из претендентов. Начиная со следующего студента, процедура повторяется. Последний оставшийся студент становится старостой. Ввести значения *M, N, K* и найти номер старосты.

20. В символьной строке записано выражение из цифр и операций в постфиксной форме (знак операции после операндов). Проверить правильность записи и найти значение выражения с помощью стека.

Пример: выражение (2+3)·4−5·6 записывается в постфиксной форме как 23+4·56·−.

21. Текст программы на бейсике содержит циклы вида

FOR ID= .... # NEXT ID ,

где # − числовой номер (метка);

ID − идентификатор параметра цикла.

Требуется:

а) проверить правильность идентификаторов, задающих параметры циклов (не более двух символов, первый из которых − латинская буква, а второй − цифра);

б) с помощью стека проверить правильность вложенности циклов;

в) переписать файл с исходным текстом так, чтобы операторы цикла каждого последующего уровня вложения были сдвинуты на две позиции вправо по сравнению с предыдущим уровнем.

22. Организовать в основной памяти стек из очередей. Обеспечить операции ведения очереди из вершины стека, расширения и сокращения стека, выдачи содержимого стека.

23. Автостоянка содержит одну полосу, на которой может быть размещено до N машин. Имеется план прибытия и убытия машин на стоянку. Если в момент прибытия стоянка оказывается занятой, машина уезжает восвояси. Машины въезжают с южной стороны, а могут выехать из K ($K<N$) крайних мест северной стороны. При выезде машины очередь сдвигается с юга на север.

Требуется:

а) промоделировать работу стоянки, последовательно сообщая о приходе и убытии машин с выдачей информации о стоящих машинах;

б) выдать два списка номеров машин: тех, которые не смогли встать на стоянку, и тех, что не смогли выехать вовремя.

24. Каким образом можно реализовать очередь из стеков? Стек из очередей? Очередь из очередей? Написать программу, реализующую соответствующие операции для каждой из рассмотренных структур данных.

25. Привести четыре способа реализации очереди очередей при помощи списка и очередей, реализованных на базе массива. Напишите для каждой реализации следующие программы:

− удаление очереди из очереди очередей *qq* и присваивает ее *q*;

− помещает очередь *q* в *qq*;

− удаляет элемент из первой очереди в *qq* и присваивает его x;

− помещает элемент *x* в первую очередь *qq*.

Определить аналогичные операции для стека из стеков, и стека из очередей, а также для очереди из стеков.

26. *Очередь с двусторонним доступом* – список, в котором добавлять и удалять элементы можно с обоих концов. Разработать реализации для таких очередей с использованием массивов, указателей и курсоров.

27. Как реализовать очередь, если элементами являются символьные строки произвольной длины? Сколько времени необходимо для операции вставки такого элемента в очередь?

28. В одной возможной реализации очередей посредством связанных списков не используется ячейка заголовка, а указатель

front указывает непосредственно на первую ячейку. Если очередь пуста, тогда *front = rear = nil*. Написать необходимые операторы для этой реализации очередей.

29. В одном варианте реализации очередей посредством циклических массивов записываются позиция первого элемента и длина очереди.

Необходимо ли в этой реализации ограничивать длину очереди числом *maxlength* – 1?

Напишите пять операторов, выполняемых над очередями, для этой реализации.

30. Возможно хранение двух стеков в одном массиве, если один располагается в начале массива и увеличивается к концу массива, а второй располагается в конце массива и увеличивается к началу. Напишите процедуру PUSH(*x, S*) вставки элемента *x* в стек *S*, где *S* – один или другой стек из этих двух стеков. Включить все необходимые проверки в эту процедуру.

31. Можно хранить *k* стеков в одном массиве, если используется структура данных, показанная на рис.1 для $k = 3$. Тогда можно организовать вставку и удаление элементов для каждого стека. Однако если при вставке элемента в стек *i* вершина TOP(*i*) совпадет с «дном» предыдущего стека BOTTOM(*i*–l), то возникнет необходимость переместить все стеки так, чтобы между каждой парой смежных стеков был зазор из пустых ячеек массива. Тогда мы можно сделать все зазоры одинаковыми или пропорциональными длине соседнего с зазором стека (из теории следует: чем больше стек, тем вероятнее в ближайшем будущем его увеличение, а мы, естественно, хотим отсрочить следующую реорганизацию массива).

32. Написать код для пяти операторов стека в предположении, что есть процедура *reorganize* (реорганизация), вызываемая при возникновении «конфликтов» между стеками.

33. Написать процедуру *reorganize* в предположении, что существует процедура *makenewtops* (сделать новые вершины), которая вычисляет *newtop[i],* – новую позицию вершины стека *i* для всех *i*, $1 \le i \le k$. *Совет:* отметим, что стек *i* может перемещаться как вверх, так и вниз по массиву. Если новая позиция стека *j* захватывает

старую позицию стека *i*, то стек *i* необходимо переместить раньше стека *j*. Рассматривая стеки в порядке 1, 2, ..., *k,* создадим еще стек «целей», каждая «цель» будет перемещать отдельный конкретный стек. Если стек *i* можно безопасно переместить, то перемещение выполняется и повторно рассматривается стек, чей номер находится в вершине стека целей. Если стек *i* нельзя переместить, не затрагивая других стеков, то его номер помещается в стек целей;

34. Какая подходящая реализация для стека целей? Необходимо ли для этого использовать список из целых чисел или можно воспользоваться более простой реализацией?

35. Реализовать процедуру *makenewtops* так, чтобы пространство перед каждым стеком было пропорционально его текущей длине.

36. Какие изменения необходимы в схеме рис. 1, чтобы ее можно было применить для очередей? А для произвольных списков?

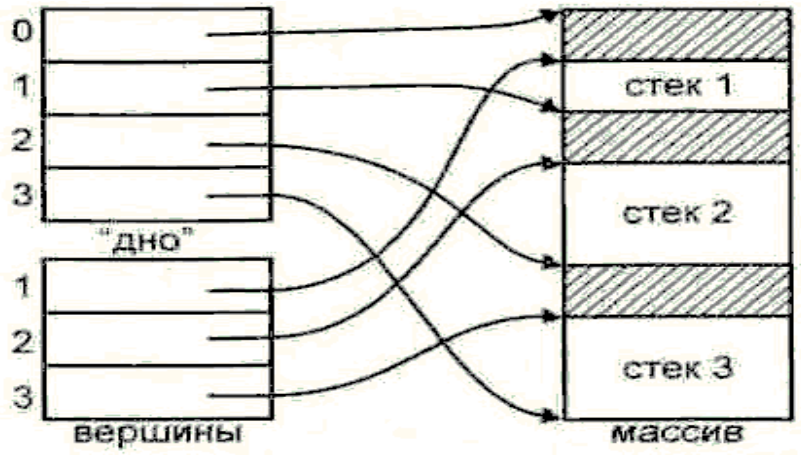

Рис. 1. Расположение, нескольких стеков в одном массиве

37. Задача Джозефуса. Отряд солдат окружен превосходящими силами противника. Надежда на победу без подкрепления исключается, однако для прорыва из лагеря имеется одна лошадь. Солдаты решают выбрать одного человека и послать его за помощью. Они становятся в круг и из шляпы выбирается число *n* и одно из их имен. Производится счет по часовой стрелке по кругу, начиная с солдата с выбранным именем. Когда счетчик достигает *n,* соответствующий солдат удаляется из круга, а счет продолжается снова, начиная со следующего солдата. Последний оставшийся в

круге солдат посылается за помощью. Определить порядок удаления солдат из круга и имя оставшегося солдата.

33. Считалка. Группа людей стоит в кругу и каждый выбирает целое положительное число. Затем выбираются одно из их имен и положительное число *n*. Производится счет по часовой стрелке, начиная с человека с выбранным именем. При этом *n*-й человек исключается из круга. Выбранное этим человеком число используется для продолжения счета. Эти действия повторяются до тех пор, пока из всего круга остается только один человек. Определить порядок удаления людей из круга и имя оставшегося человека.

34. Для работы с очередью, т. е. последовательностью элементов, в которую элементы всегда добавляются в конец, а удаляются из начала («первым пришел—первым ушел»), нужны обычно следующие операции»

04ИСТ0Ч(Q)—создать пустую очередь *Q* (очистить очередь);

ПУСТ0Ч(Q)—проверить, является ли очередь *Q* пустой;

В0ЧЕРЕДЬ(Q,x)—добавить в конец очереди *Q* элемент *x;*

ИЗОЧЕРЕДИ(Q,x)—удалить из очереди *Q* первый элемент, присвоив его параметру *x.*

Требуется для каждого из указанных ниже представлений очереди описать на Паскале соответствующий тип *очередь,* считая, что все элементы очереди имеют некоторый тип *ТЭО,* и реализовать в виде процедур или функций перечисленные операции над очередью (если операция по тем или иным причинам не может быть выполнена, следует передать управление некоторой процедуре *ОШИБ КА(k)*, считая ее уже описанной, где *k*—номер ошибки: 1—переполнение очереди, 2—исчерпание очереди).

Представление очереди *(n*—целая константа>1):

а)* для каждой очереди отводится свой массив из *n* компонент типа *ТЭО,* в котором элементы очереди занимают группу соседних компонент, индексы первой и последней из которых запоминаются (рис. 16, а); при этом, когда очередь достигает правого края массива, все ее элементы сдвигаются к левому краю;

б) аналогичное представление, но массив как бы склеивается в кольцо, поэтому если очередь достигает правого края массива, то новые элементы записываются в начало массива (рис. 16,б);

в) для каждой очереди создается свой однонаправленный список из элементов типа *ТЭО,* при этом запоминаются ссылки на первое и последнее звенья списка (рис. 16, в*).*

35. Используя очередь (считать уже описанными тип *очередь* при подходящем типе *ТЭО,* функцию *ПУСТОЧ* и процедуры *ОЧИСТОЧ, ВОЧЕРЕДЬ* и *ИЗОЧЕРЕДИ*—см. 17.1), решить следующую задачу (решение описать в виде процедуры).

а)* type FR=file of real;

За один просмотр файла *f* типа FR и без использования дополнительных файлов напечатать элементы файла *f* в следующем порядке: сначала—все числа, меньшие я, затем—все числа из отрезка [a,b], и наконец—все остальные числа, сохраняя исходный взаимный порядок в каждой из этих трех групп чисел *(a* и *b*—заданные числа, *a<b).*

б) Содержимое текстового файла *f*, разделенное на строки, переписать в текстовый файл *g*, перенося при этом в конец каждой строки все входящие в

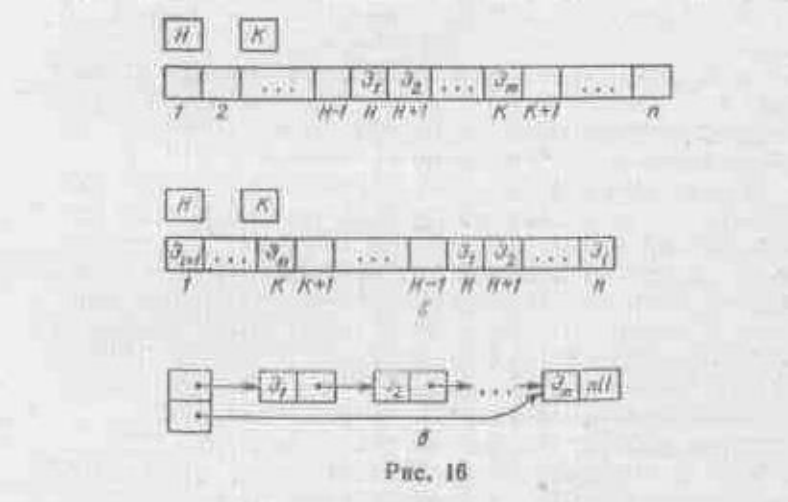

Рис. 16

нее цифры (с сохранением исходного взаимного порядка как среди цифр, так и среди остальных литер строки),

в) type имя = (Анна,....Яков);
дети = packed array [имя, имя]
of boolean;
потомки = file of имя;

Считая заданными имя *И* и массив *Д* типа *дети (Д[x,y]=true,* если человек по имени *y* является ребенком человека по имени *x),* записать в файл *П* типа *потомки* имена всех потомков человека с именем *И* в следующем порядке: сначала—имена всех его детей, затем—всех его внуков, затем—всех правнуков и т. д.

36. Для работы со стеком, т. е. последовательностью элементов, в которой элементы всегда добавляются в конец и удаляются из конца («последним пришел—первым ушел»), нужны обычно следующие операции:

ОЧИСТЕК(S)—создать пустой стек *S* (очистить стек);

ПУСТЕК(S)—проверить, является ли стек *S* пустым;

ВСТЕК(S,x)—добавить в конец стека *S* элемент *x*;

ИЗСТЕКА (S,x)—удалить из очереди *S* последний элемент, присвоив его параметру *x.*

Требуется для каждого из указанных ниже представлений стека описать на Паскале соответствующий тип *стек,* считая, что все элементы стека имеют некоторый тип *ТЭС,* и реализовать в виде процедур или функций перечисленные операции над стеком (если операция по тем или иным причинам невыполнима, следует передать управление некоторой процедуре *ОШИБКА(k),* считая ее уже описанной, где *k*—номер ошибки» 1— переполнение стека 2— исчерпание стека).

Представление стека *(n*—целая константа>1):

а) для каждого стека отводится свой массив из *n* компонент типа *ТЭС,* в начале которого располагаются элементы стека, при этом запоминается индекс компоненты массива, занятой последним элементом стека (рис. 17, а*):*

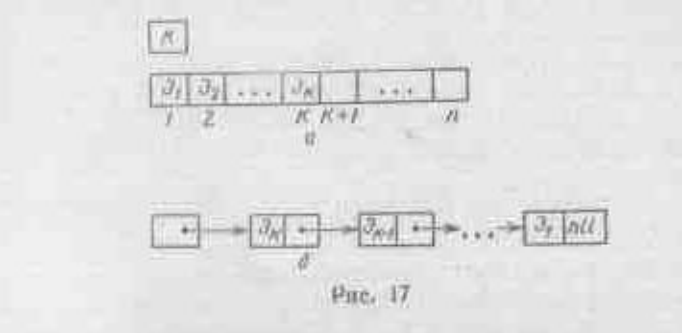

Рис. 17

б) для каждого стека создается свой однонаправленный список, в котором элементы стека располагаются в обратном порядке (рис. 17,б).

37. Используя стек (считать уже описанными тип *стек* при *ТЭС—char,* функцию *ПУСТЕК* и процедуры *ОЧИСТЕК, ВСТЕК* и *ИЗСТЕКА*— см. 17.3), решить следующую задачу (решение описать в виде процедуры или функции).

а) Напечатать содержимое текстового файла *t*, выписывая литеры каждой его строки в обратном порядке.

б) Проверить, является ли содержимое текстового файла *t* правильной записью формулы следующего вида:

<формула>:: — <терм> | <терм>+<формула> |
<терм>—<формула>
<терм>::=<имя> |(<формула>) | [<формула>]|
{<формула>}
<имя>::=x | y | z

в)* В текстовом файле *f* записана без ошибок формула следующего вида:

<формула>::=<цифра> | M(<формула>,<формула>) |
m<формула>,<формула>)
<цифра>::=0|1|2|3|4|5|6|7|8|9

где M обозначает функцию max, а m—min.

Вычислить (как целое число) значение данной формулы (например, M(5,m(6,8))→6).

г) В текстовом файле LOG записано без ошибок логическое выражение (ЛВ) в следующей форме:

<ЛВ>::= true | false | (┐ <ЛВ>) | (<ЛВ>^ <ЛВ>) |
(<ЛВ>ˇ <ЛВ>)

где знаки ┐, ^ и ˇ обозначают соответственно отрицание, конъюнкцию и дизъюнкцию.

Вычислить (как *boolean)* значение этого выражения.

38. Используя очередь и/или стек (считать уже описанными их типы и операции над ними—см. 17.1 и 17.3), решить следующую задачу (решение описать в виде процедуры).

R текстовом файле *t* записан текст, сбалансированный по круглым скобкам:

<текст>:: = <пусто> | <элемент><текст>

<элемент>::=<буква> |(<текст>)

Требуется для каждой пары соответствующих открывающей и закрывающей скобок напечатать номера их позиций Р тексте, упорядочив пары номеров в порядке возрастания номеров позиций:

а) закрывающих скобок;

б) открывающих скобок.

Например, для текста A+(45—F(X)*(B—C)) надо напечатать:

а) 8 10; 12 16; 3 17;

б) 3 17; 8 10; 12 16.

39. Под «выражением» будем понимать конструкцию следующего вида:

< выражение>::=<терм>|
<терм> <знак+—> <выражение>

<знак+—>::= + | —

<терм>::=<множитель> | <множитель>*<терм>

<множитель>::=<число> | <переменная> | (<выражение>) |
<множитель> ↑ <число>

<число>::=<цифра>

<переменная>::=<буква>

где знак ↑ обозначает операцию возведения в степень.

Постфиксной формой записи выражения aΔЬ называется запись, в которой знак операции размещен за операндами: abΔ. Примеры:

a—b → *ab—*

*a*b+c* → *ab*+* (т.е. *(ab*)c+)*

a(b+c)* → *abc+** (т.е. *a(bc+)*)*

*a+b ↑c ↑d*e* →* *abc ↑ d ↑*e* +

а) Описать функцию *vatue(postfix],* которая вычисляет как целое число значение выражения (без переменных), записанного в постфиксной форме в текстовом файле *postfix.*

Использовать следующий алгоритм вычисления. Выражение просматривается слева направо. Если встречается операнд (число), то его значение (как целое) заносится в стек, а если встречается знак операции, то из стека извлекаются два последних элемента (это операнды данной операции), над ними выполняется операция и ее результат записывается в стек. В конце концов в стеке останется только одно число—значение всего выражения.

б) Описать процедуру *translate(infix,postfix),* которая переводит выражение, записанное в обычной (инфиксной) форме в текстовом файле *infix,* в постфиксную форму и в таком виде записывает его в текстовый файл *postfix.*

Использовать следующий алгоритм перевода. В стек записывается открывающая скобка, и выражение просматривается слева направо. Если встречается операнд (число или переменная), то он сразу переносится в файл *postfix.* Если встречается открывающая скобка, то она заносится в стек, а если встречается закрывающая скобка, то из стека извлекаются находящиеся там знаки операций до ближайшей открывающей скобки, которая также удаляется из стека, и все эти знаки (в порядке их извлечения) записываются в файл *postfix.* Когда же встречается знак операции, то из конца стека извлекаются (до ближайшей скобки, которая сохраняется в стеке) знаки операций, старшинство которых больше или равно старшинству данной операции, и они записываются в файл *postfix,* после чего рассматриваемый знак заносится в стек. В заключение выполняются такие же действия, как если бы встретилась закрывающая скобка.

в) Описать (нерекурсивную) процедуру *infixprint(postfix),* которая печатает в обычной (инфиксной) форме выражение, записанное в постфиксной форме в текстовом файле *postfix.* (Лишние скобки желательно не печатать.)

6. ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Составить программу работы с циклической очередью, организованной с помощью структуры с указателями. Обеспечить операции постановки в очередь, продвижения очереди, вставки в середину перед элементом с заданным ключом и удаления из середины.

```
    Program CiclList;
    Uses crt;
    Type Ukaz=^List;
List = Record
Key : Integer;
Next : Ukaz;
```

```
End; {кольцевая очередь; указатели от начала к концу}
{конечный элемент замыкается на начало} Var p, q, r:
Ukaz; {r - указатель на конечный элемент}
    Knew, k, tmp, CouElem : Integer;
    CouElem - число элементов в очереди { Постановка в
очередь }
    Procedure MakeList;
    Begin
    While True Do
Begin
Write ('Ключ: ') ;
ReadLn (Knew) ;
If Knew = 0 Then
Begin
WriteLn ('Всего элементов: ', CouElem) ;
Exit
End;
New (q);
If CouElem=0 Then
q^.next :=q {первый элемент указывает сам на себя}
Else
Begin
q^.next :=r^.next; {r^.next --> начало очереди}
r^.next :=q {связь с бывшим концом очереди}
End;
    q^.key :=Knew;
    r: =q {новый элемент}
    Inc (CouElem) ;
    End
    End;
{ Продвижение очереди }
    Procedure Go;
    Var Count : Integer;
    Begin
    p: =r^.next; {указатель на начало очереди}
    WriteLn ('Ключ начальной позиции', p^key) ;
    Dec (CouElem) ;
    If CouElem>0 Then
    begin
    q: =p^.next; {второй элемент}
    r^.next: =q;
    WriteLn ('Ключ новой начальной позиции', q^.key)
    end
    Else WriteLn ('Очередь пуста') ;
    Dispose (p)
```

```
    End;
{ Вставка в середину перед элементом с заданным ключом }
    Procedure IncList;
    Var c: Integer;
g: Ukaz;
    Begin
    c: =0;
    Write ('Перед элементом с ключом') ;
    ReadLn (k);
    p: =r^.next; {начало очереди}
    Repeat
    If p^.key = k Then {ищем элемент с нужным ключом}
    Begin
    New (g); {и вставляем перед ним новый}
    g^: =p^;
    p^.next: =g;
    Write ('Ключ: ') ;
    ReadLn (Knew) ;
    If Knew = 0 Then Exit;
    p^.key: =Knew;
    If c=CouElem-1 Then r: =g; {вставка перед концом}
    Inc (CouElen) ;
    Exit
    End
    Else
    Begin
    p :=p^.next; {продвинуться по очереди}
    Inc (c)
    End
    Until  c=CouElem;  {цикл  до  возврата  в  исходную
позицию}
    WriteLn ('В списке элемента с ключом ', k, 'нет')
    End;
{ Удаление из середины очереди }
    Procedure DelFromList;
    Var c : Integer;
    Begin
    c: =0;
    Write ('С каким ключом?') ;
    ReadLn (k);
    p: =r^.next; {начало очереди}
    Repeat
    If p^.key = k Then
    Begin {ищем элемент с нужным ключом}
    If p = r^.next Then Go {удаление из начала очереди}
```

```
    Else
    Begin
    q^.next := p^.next;
    If p=r Then r: =q; {конец очереди}
    Dispose (p);
    Dec (CouElem) ;
    End;
    Exit;
    End
    Else
    Begin
    q: =p; {запомнить предыдущий элемент}
    p: =p^.next; {продвинуться по очереди}
    Inc (c)
    End
    Until  c=CouElem;  {цикл  до  возврата  в  исходную
позицию}
    WriteLn ('В списке элемента с ключом', k, 'нет') ;
    End;
{ Выдача очереди на экран }
    Procedure OutList;
    Var Count, i : Integer;
    Begin
    p: =r^.next; {начало очереди}
    For i: =1 To CouElem Do
    Begin
    Write (p^.key, ' ') ;
    p: =p^.next
    end;
    WriteLn;
    WriteLn ('Всего элементов: ', CouElem)
    End;
 { Головная программа }
Begin
    Clrscr;
    CouElem: =0; {счетчик числа элементов в очереди}
    Repeat
    WriteLn;
    WriteLn  ('1.  Поставить  в  очередь  (ключ  0  -  признак
конца) ') ;
    WriteLn ('2. Продвинуть очередь') ;
    WriteLn ('3. Вставить элемент') ;
    WriteLn ('4. Удалить элемент') ;
    WriteLn ('5. Показать всю очередь') ;
    WriteLn ('6. Выход') ;
```

```
    Write ('Выбираем :') ;
    ReadLn (Tmp) ; {номер пункта меню}
    Case Tmp Of
1: MakeList;
2: If CouElem <> 0 Then Go
Else WriteLn ('Очереди нет, повторите выбор') ;
3: If CouElem <> 0 Then IncList
Else WriteLn ('Очереди нет, повторите выбор') ;
4: If CouElem <> 0 Then DelFromList
Else WriteLn ('Очереди нет, повторите выбор') ;
5: If CouElem <> 0 Then OutList
Else WriteLn ('Очередь пуста') ;
6: Halt
    End
    End.
```

7. УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Конечным результатом выполнения задач в лабораторной работе является отлаженная программа. Текст программы представляется на машинных носителях и должен содержать постановку задачи, сведения об авторе и подробные комментарии к выполняемым операторам программы.

После выполнения лабораторной работы преподаватель проверяет качество оформления текста программы и правильность функционирования программы сначала на тестах автора, а затем и на других данных. Все используемые обозначения должны быть расшифрованы.

Текст программы должен содержать подробные комментарии поясняющие назначения процедур, их параметры, использование переменных, смысл и особенности реализации отдельных программных блоков.

В разделе «Тестирование и отладка» привести перечень тестов и описание результатов, полученных на этих тестах. Если полученные результаты не очевидны, необходимо привести их ручной расчет и обоснование.

Преподаватель вправе задать вопросы по всем разделам отчета, в частности по программным фрагментам, поэтому, качественное документирование программы способствует успешной защите работы.

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что собой представляет структура типа стек?
2. Как реализуется структура типа стек в различных алгоритмических языках?
3. Когда применяется структура типа стек?
4. Чем отличается стек от массива?
5. Какие преимущества имеет структура типа стек?
6. Как вставляются и удаляются элементы из стека?
7. Что собой представляет структура типа очередь?
8. Как реализуется структура типа очередь в различных алгоритмических языках?
9. Когда применяется структура типа очередь?
10. Чем отличается очередь от массива?
11. Какие преимущества имеет структура типа очередь? Привести примеры.
12. Как вставляются и удаляются элементы из очереди?
13. Что такое циклическая очередь?
14. Как выбирается размер стека?
15. Как выбирается размер очереди?
16. Что такое очередь с приоритетами?
17. В чем заключается суть стековых машин?
18. Что такое структура типа дек?
19. Что такое линейный список?
20. Можно ли реализовать структуры типа стек, очередь, дек с помощью линейных списков?
21. Почему структура типа дек применяется редко?
22. В чем преимущества и недостатки организации структур в виде стека?
23. В чем заключаются преимущества и недостатки организации структур в виде очереди?
24. Для моделирования каких реальных задач удобно использовать стек? А для каких очередь?
25. Какое значение хранит указатель на стек?
26. Какое значение хранит указатель на очередь?
27. Какие существуют ограничения на тип информационного поля стека и очереди?
28. С какой целью в программах выполняется проверка на пустоту стека и очереди?

29. При работе со стеком или очередью доступны позиции ограниченного числа элементов. Возможна ли ситуация записи новых элементов стека или очереди на уже занятые собственными элементами участки памяти (запись себя поверх себя)? Ответ обоснуйте.

30. С какой целью в программах выполняется удаление стека и очереди по окончании работы с ними? Как изменится работа программы, если операцию удаления не выполнять?

9. СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Трамбле Ж. Ведение в структуры данных / Ж. Трамбле, П. Соренсон. – М.: Машиностроение, 1982. – 784 с.

2. Топп, У. Структуры данных в С ++: пер. с англ / У. Топп , У. Форд. – М.: БИНОМ, 1999. – 816 с.

3. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М: МЦНМО, 1999. –960 с.

4. Ахо, А. В. Структуры данных и алгоритмы / А. В. Ахо, Д. Э. Хопкрофт, Д. Д. Ульман. – М.: Вильямс, 2000. – 384 с.

5. Кубенский, А. А. Создание и обработка структур данных в примерах на Java/А. А.Кубенский – СПб. БХВ-Петербург, 2001. – 336 с.

6. Хусаинов, Б. С. Структуры и алгоритмы обработки данных. Примеры на языке Си: учеб. пособ / Б. С Хусаинов – Финансы и статистика, 2004. – 464 с.

Структуры и алгоритмы обработки данных. Очереди и стеки: методические указания к выполнению лабораторной работы №3 для студентов очной, очно-заочной и заочной форм обучения по направлениям подготовки 230100 «Информатика и вычислительная техника», 010500 «Математическое обеспечение и администрирование информационных систем», 231000 «Программная инженерия»

ВАСИЛИЙ КОНСТАНТИНОВИЧ ГУЛАКОВ

Научный редактор В.В. Конкин
Редактор издательства Л.Н. Мажугина
Компьютерный набор В.К. Гулаков

Темплан 2013 г., п.219

Подписано в печать Формат 1/16 Бумага офсетная. Офсетная печать. Усл.печ.л. 1,52 Уч.-изд.л. 1,52 Тираж 40 экз. Заказ Бесплатно

Издательство Брянского государственного технического университета
241035, Брянск, бульвар им.50-летия Октября, 7, БГТУ, тел. 58-82-49
Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16