МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

А. В. СОНИЧЕВ

СТРУКТУРНЫЙ РЕФАКТОРИНГ ДИАГРАММ BPMN

Учебно-методическое пособие

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2022

УДК 004.43(07)
ББК З 973.2-018.1я7
С62

Соничев А. В.

С62 Структурный рефакторинг диаграмм BPMN: учебно-методическое пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2022. 46 с.

ISBN 978-5-7629-

Содержит описания преобразований алгоритмов оркестрованных бизнес-процессов к виду, доступному для практической реализации на языке BPEL в рамках архитектуры SOA. Для представления алгоритмов используются традиционные блок-схемы; для проверки свойств алгоритмов – синтаксические структуры алгоритмического языка **C** (Си); для проверки эквивалентности алгоритмов – ручное тестирование методом «белого ящика»; для представления структуры тегов BPEL – схемы Дамке и Насси–Шнейдермана. Для графического представления бизнес-процессов используется нотация BPMN Aris Express.

Предназначено для подготовки бакалавров по направлению 09.03.02 – «Информационные системы и технологии» и специалистов по специальности 090301.65 «Компьютерная безопасность».

УДК 004.43(07)
ББК З 973.2-018.1я7

Рецензент кафедра информационных систем и технологий Факультета информатики и прикладной математики Федерального государственного бюджетного образовательного учреждения высшего образования «Санкт-Петербургский государственный экономический университет».

Утверждено
редакционно-издательским совета университета
в качестве учебно-методического пособия

ISBN 978-5-7629

ЦЕЛЬ РАБОТЫ

Средством реализации оркестрованных бизнес-процессов (БП) [1] в рамках архитектуры SOA (**S**ervice-**O**riented **A**rchitecture – сервис-ориентированная архитектура) [2] является язык BPEL (**B**usiness **P**rocess **E**xecution **L**anguage – язык выполнения бизнес–процессов) [3]. Оркестровка предполагает наличие некоторого координатора, который привлекает участников БП ***по заданному алгоритму*** и определяет последовательность вызова web-сервисов. [4]. Язык BPEL построен на основе языка XML (*e**X**tensible **M**arkup **L**anguage* – расширяемый язык разметки). Алгоритм БП описывается управляющими операторами, содержащимися в тегах: *sequence* (последовательность выполнения действий); *switch* (выбор); *while* (цикл) [5]. Таким образом, в BPEL отсутствуют операторы передачи управления, например такие как *break*, *continue* и *goto* в языке **C**(Си), поэтому в BPEL могут быть реализованы только структурные алгоритмы – алгоритмы, не содержащие явных передач управления [6].

При проектировании и разработке БП используются диаграммы BPMN [7], допускающие автоматическую генерацию кода на BPEL, при этом, естественно, алгоритмы БП должны быть структурными и строиться по правилам структурного программирования – последовательной декомпозицией задач БП [8].

В БП реальных предприятий, как правило, задействованы несколько подразделений, а возможно и подразделения других предприятий. При этом практически неизбежны различные переходы от решения одних задач к другим в зависимости от сложившихся условий, и построение алгоритма БП по правилам структурного программирования является весьма проблематичным. Поэтому процесс разработки диаграмм BPMN после «добавления развилок и других событий» [9] должен содержать преобразование алгоритма БП к эквивалентному алгоритму структурного вида, в противном случае неизбежно появление семантических ошибок в процессе его выполнения в BPEL [10].

Преобразование алгоритма к структурному виду существует всегда в соответствии с теоремой Бома–Якопини (Bohm Corrado and Giuseppe Jacopini*:* «любой исполняемый алгоритм может быть преобразован к структурному виду, т.е. такому виду, когда ход его выполнения определяется только при помощи трёх структур управления: последовательной, ветвлений и повторов или циклов» [11].

Целью работы является формирование навыков преобразования алгоритмов к структурному виду и разработки диаграмм BPMN по заданным алгоритмам БП.

ЗАДАНИЕ

1. Преобразовать заданный вариант блок–схемы алгоритма БП к структурному виду без цикла с постусловием (использовать метод введения переменной состояния не рекомендуется).

2. Доказать, что полученный алгоритм является структурным (синтаксическая структура программы, реализующей этот алгоритм, не должна содержать операторов передачи управления).

3. Доказать эквивалентность заданного и преобразованного алгоритмов и соответствие блок-схем синтаксическим структурам программ методом «белого ящика».

4. Представить работу алгоритма преобразования последовательностью схем Дамке.

5. Разработать схемы Насси–Шнейдермана структуры описания БП на языке BPEL.

6. Построить диаграммы BPMN оркестрованных БП в среде ARIS Express для заданного и эквивалентного алгоритма, допускающего реализацию на языке BPEL в рамках архитектуры SOA.

1. ПОРЯДОК ВЫПОЛНЕНИЯ

1.1. Преобразование блок-схем алгоритмов к структурному виду

Блок-схема структурного алгоритма должна состоять из трех основных (базовых) конструкций: последовательности, альтернативного выбора *(if-then-else)* и повторение ноль или более раз *(while-do)* (

рис. 1.1) и трех дополнительных: выбора варианта (*switch),* повторения один или более раз *(do-while) и* повторение заданное количество раз *(for)* (рис. 1.2), где ***«действие» обозначает либо отсутствие какой-либо внутренней структуры, либо одну из перечисленных выше структур***. Таким образом, структуры являются вложенными друг в друга, и поэтому в программах необходимость передачи управления операторами, подобными goto, просто отсут-

ствует. Каждая структура имеет только один вход и один выход, и после выполнения входящих в нее действий осуществляется переход к выполнению структуры, в которую она входит.

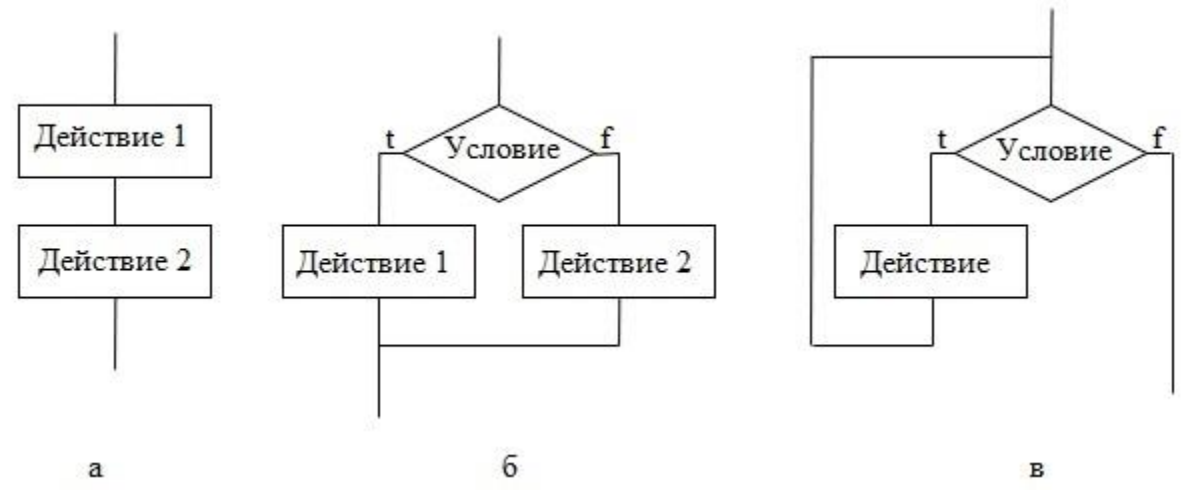


*Рис. 1.1***.** Базовые алгоритмические структуры: *а* – последовательность; *б* – выбор; *в* – цикл с предусловием

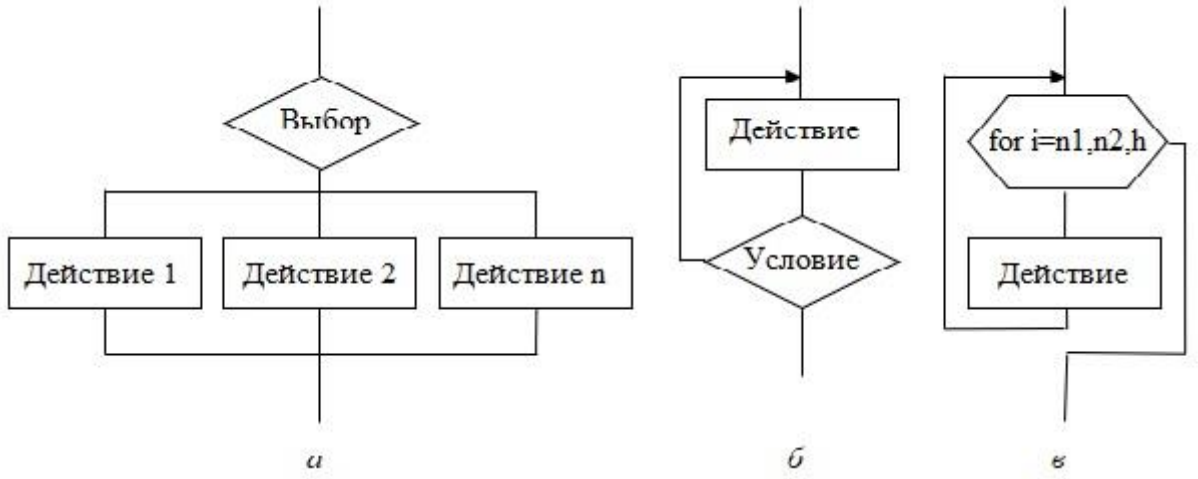


*Рис. 1.2***.** Дополнительные алгоритмические структуры: а – выбор (switch); б – цикл с постусловием (do-while); в – цикл с параметром (for)

Существует несколько методов преобразования блок схем к структурному виду.

1.1.1. Метод введения переменной состояния

Все блоки исходного алгоритма нумеруются, начиная с цифры 1, и вводится дополнительная переменная, указывающая на номер блока, который должен быть выполнен. Блок выполняется, если его номер совпадает со значением этой переменной. После выполнения блока переменной присваивается

номер очередного блока исходного алгоритма. Полученный таким образом алгоритм заканчивает свою работу, когда после выполнения последнего блока исходного алгоритма этой переменной будет присвоено значение 0.

Пример такого преобразования приведен на рис. 1.3.

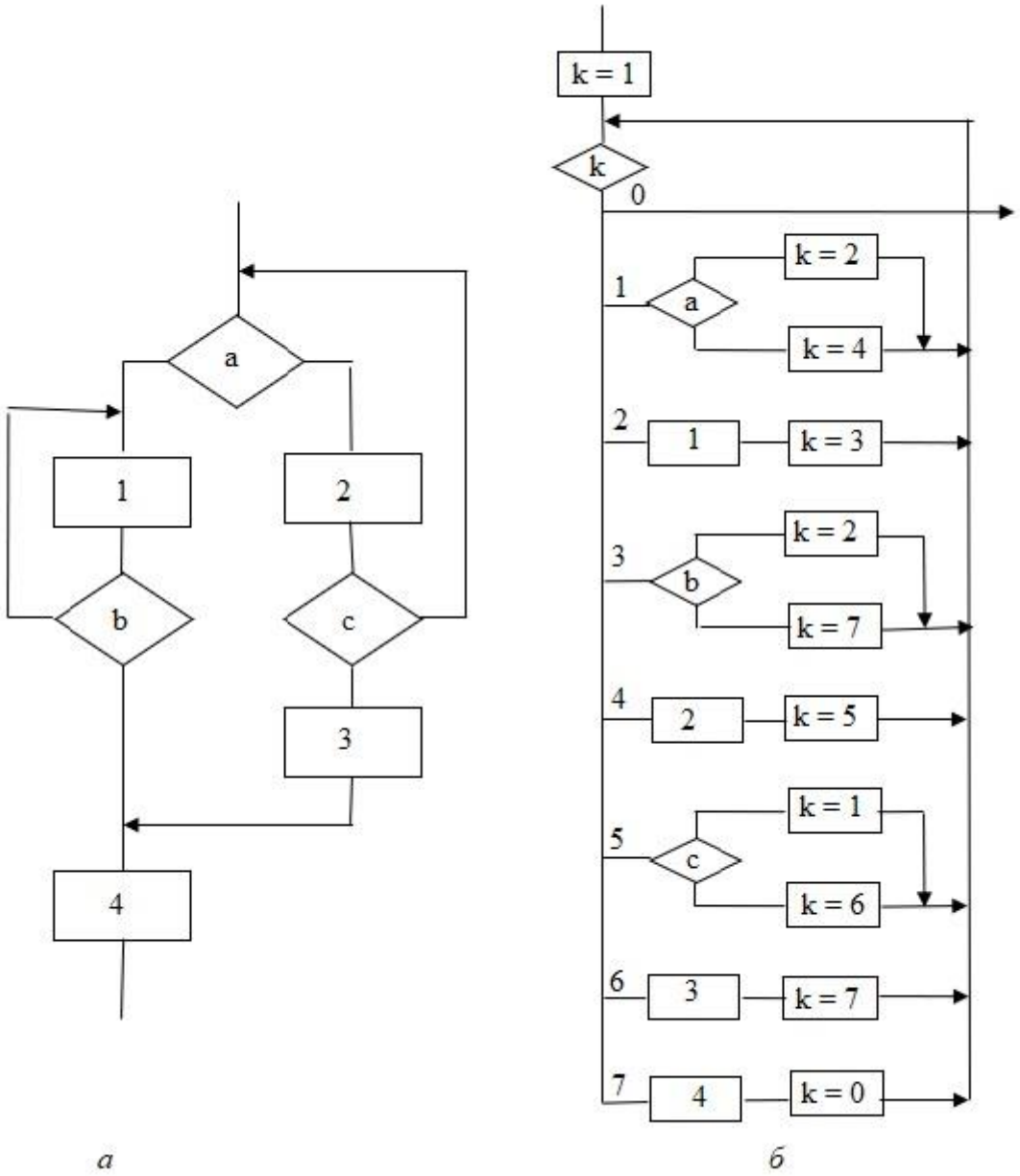


Рис. 1.3. Пример преобразования алгоритма к структурному виду методом введения переменной состояния: а – блок схема неструктурного алгоритма; б – блок схема эквивалентного структурного алгоритма

Полученная блок-схема является структурной, поскольку все элементы имеют один вход и один выход.

Такое преобразование, очевидно. совершенно неприемлемо, поскольку, во-первых, требует введения дополнительной переменной, а во-вторых, скрывает алгоритм БП, делая его практически нечитаемым.

1.1.2. Метод дублирования управляющих структур

Сквозное дублирование. Если в какую-либо структуру (включая структуру из одного элемента) входит более одной стрелки, то такую структуру достаточно продублировать (рис. 1.4), и тогда каждая структура блок-схемы будет содержать один вход и один выход (являться структурной). Очевидно, блок-схема, состоящая из структур структурных элементов, является структурной.

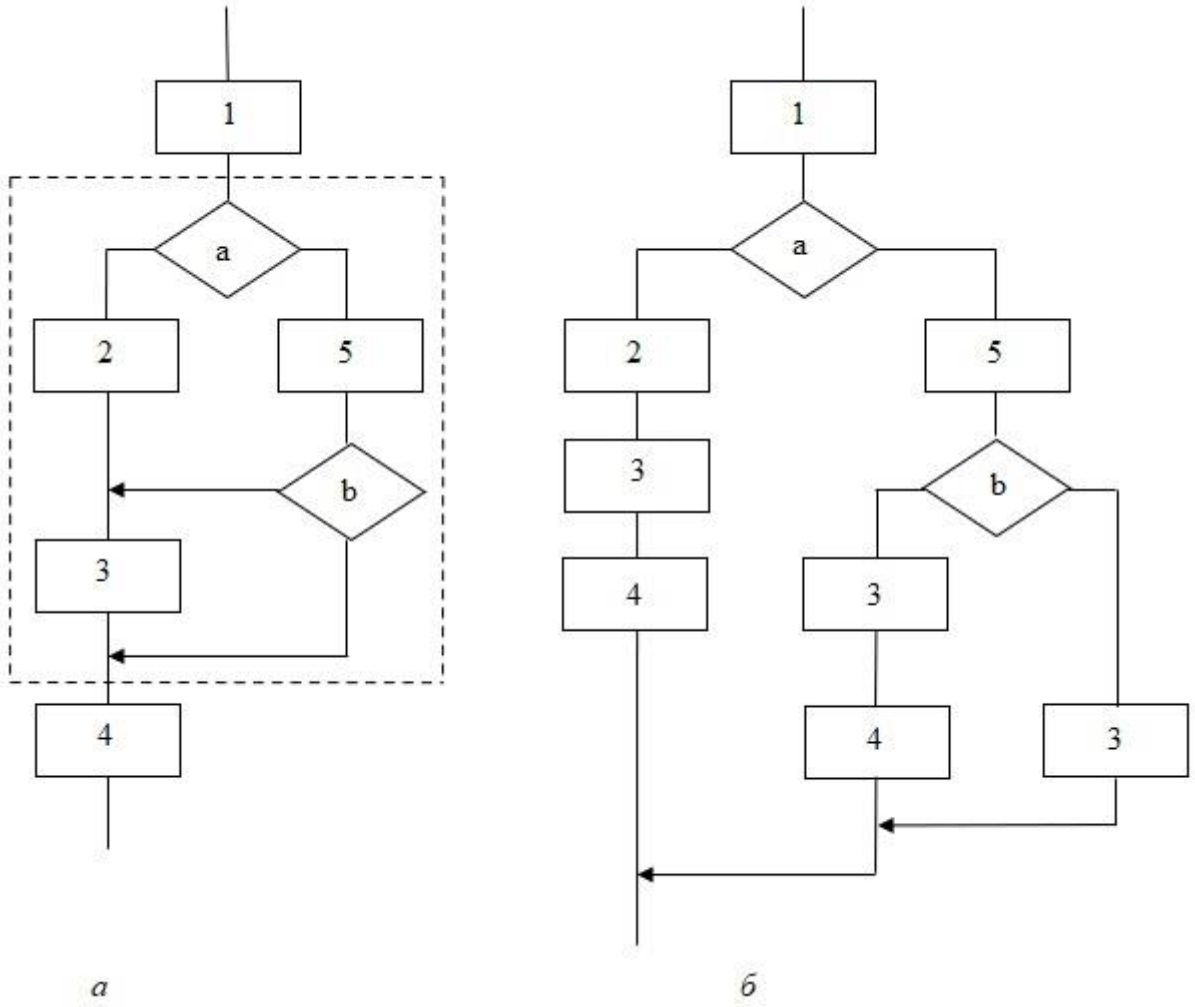


Рис. 1.4. Пример преобразования алгоритма к структурному виду методом дублирования управляющих структур: а – блок-схема неструктурного алгоритма; б – блок-схема эквивалентного структурного алгоритма

Однако такое «механическое» дублирование может привести к излишней избыточности. Например, структура выбора по «b» может быть преобразована в последовательность структур выбора и структуры, обозначенной цифрой «4» (рис. 1.5).

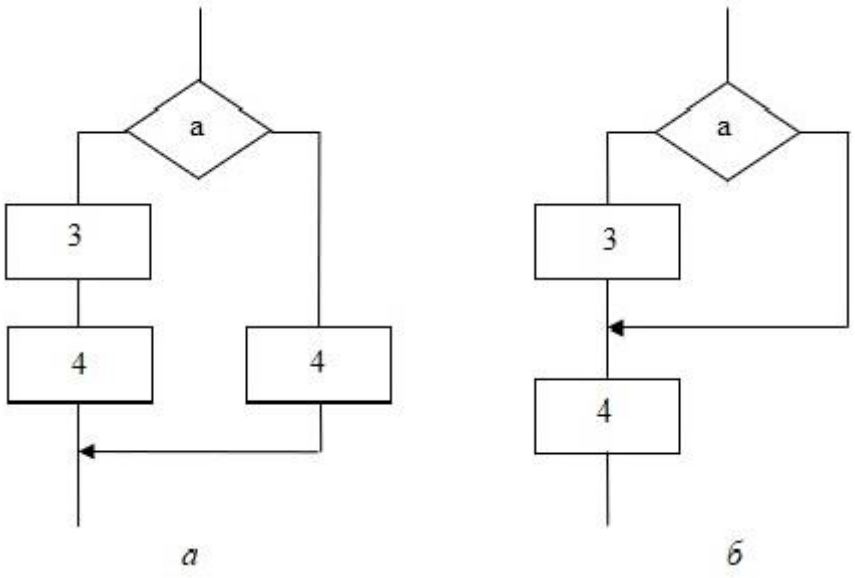


Рис. 1.5. Пример удаления избыточного дублирования: а – исходная блок-схема; б – блок-схема эквивалентного структурного алгоритма после удаления дублирования в структуре выбора по «b»

Таким образом, блок-схема на рис. 1.4,б преобразуется к блок-схеме, представленной на рис. 1.6,а, из которой аналогичным образом можно удалить «4» из выбора по «a», что и представлено на рис. 1.6,б.

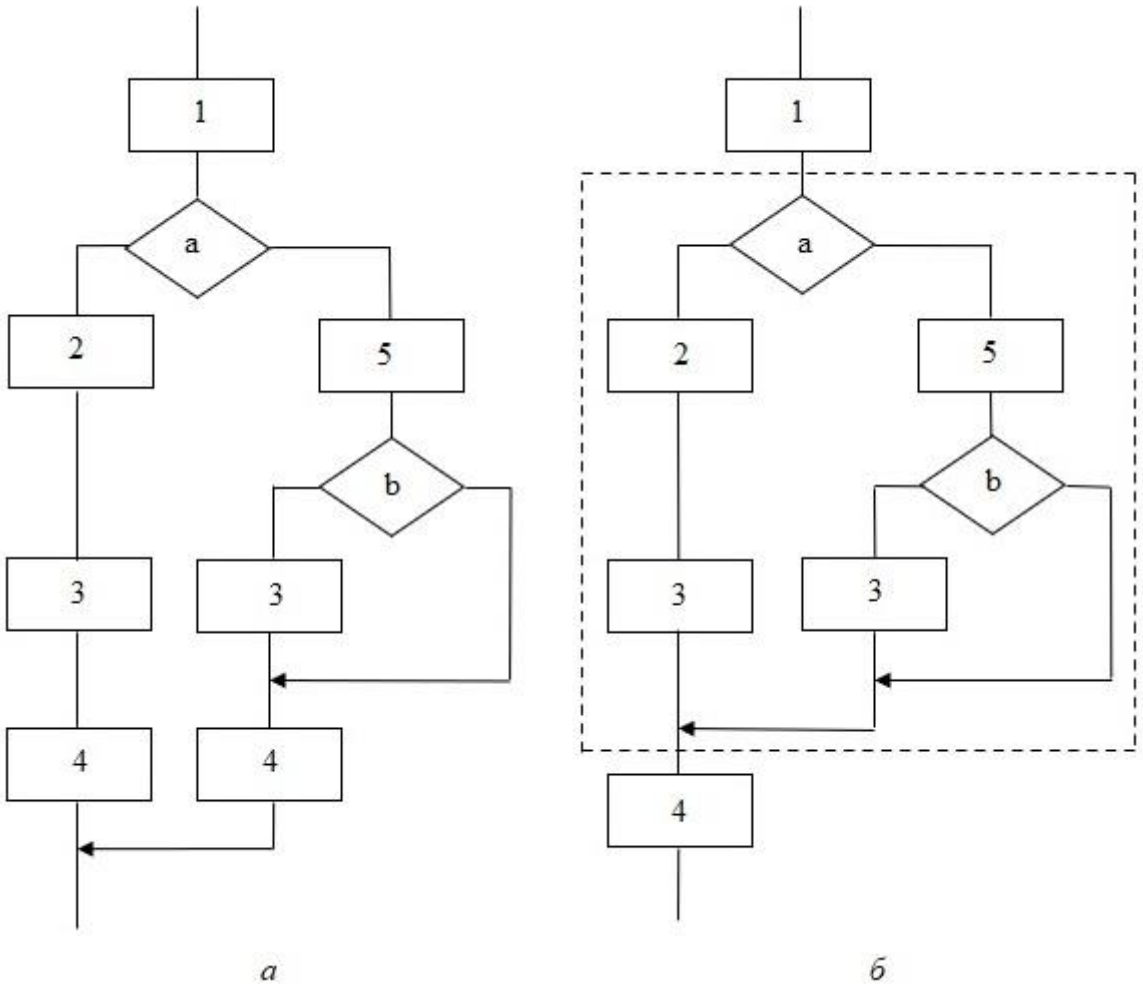


Рис. 1.6. Пример удаления избыточного дублирования: а – исходная блок-схема; б – блок-схема эквивалентного структурного алгоритма после удаления дублирования в структуре выбора по «b»

В результате исходная блок-схема преобразована к последовательности трех действий: «1»; выбор по «a» и «4», в которой выбор по «a» слева содержит последовательность «2» и «3»; справа – последовательность «5» и выбор по «b» и не содержит никаких других структур кроме последовательности и выбора. В результате продублированным оказалось только действие «3». «Совместить» это действие, подобно тому, как это было сделано для действия «4», нельзя, поскольку «3» принадлежит разным структурам: «выбор по a» и «выбор по b», в то время как «4» принадлежало одной структуре – «выбор по a».

Таким образом, используемый здесь алгоритм преобразования описывается следующим образом.

Алгоритм. Структура, содержащая более одного входа, повторяется на блок-схеме столько раз, сколько входов она имеет. Если при этом оказывается, что структура выбора содержит действия, не зависящие от значения условия

выбора, то такая структура преобразуется в структуру последовательности – сначала выбор, затем эти действия.

Внутриструктурное дублирование. Такой же результат можно получить без преобразований, представленных на рис. 1.4 – 1.6, Структурой исходной блок-схемы является последовательность «1», «выбор» (выделена на рис. 1.4,а) и «4», где «1» и «4» заведомо структурные элементы, поэтому, достаточно преобразовать только структуру «выбор», а затем подставить соответствующий ей фрагмент блок-схемы после блока «1» и добавить в последовательность блок «4».

Представленные здесь рассуждения можно обобщить на блок-схемы, содержащие вложенные неструктурные фрагменты на произвольную глубину, и тогда алгоритм преобразования блок-схем описывается следующим образом.

Алгоритм. Первоначально определяется структура исходного алгоритма. В ней выделяется неструктурный фрагмент, который принимается за исходную блок-схему, определяется его структура и выделяется его следующий неструктурный фрагмент. Процесс продолжается до тех пор, пока не будет обнаружена структура, состоящая из структурных фрагментов. Она преобразуется к структурному виду и подставляется в структуру, частью которой она являлась. Процесс продолжается до тех пор, пока исходный алгоритм не будет преобразован к структурному виду, который и представляется структурной блок-схемы, состоящей из структурных фрагментов.

1.1.3. Метод введения переменной цикла

Для преобразования блок-схем алгоритмов, содержащих ***циклы с двумя выходами***, вводится дополнительная логическая переменная. Первоначально этой переменной присваивается значение true, и она конъюнктивно добавляется в первое условие выхода. Если значение условия второго выхода равно true, выполнение цикла продолжается, в противном случае последующих действий не выполняется, и введенной переменной присваивается значение false. Таким образом, после возврата к проверке первого условия выхода из цикла он будет завершен. После выхода из цикла проверяется, по какой причине он произошел, и выполняются соответствующие действия (рис. 1.7).

Если цикл имеет ***более двух условий выхода***, то описанные выше преобразования выполняются начиная с последних условий, при этом в предыдущее условие выхода конъюнктивно добавляется переменная, связанная с предыдущим условием.

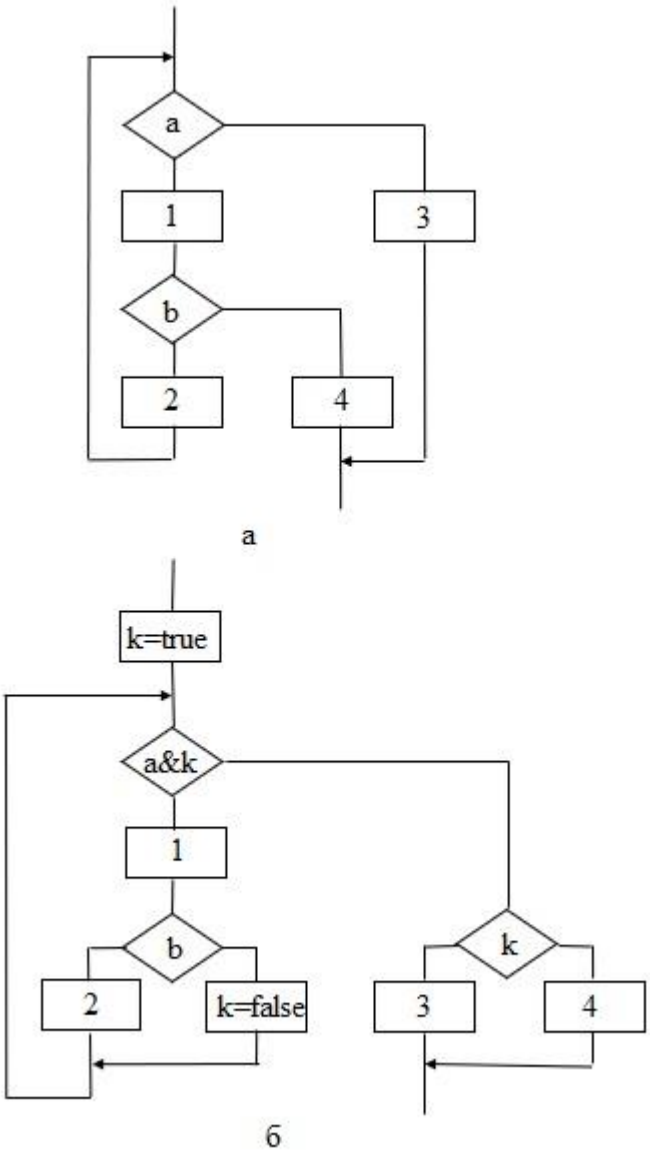


Рис. 1.7. Пример преобразования цикла, содержащего два условия выхода: Эквивалентные циклы: а – блок-схема неструктурного алгоритма; б – блок-схема эквивалентного структурного алгоритма

Частный случай этого метода представлен на рис. 1.8. Такое преобразование называется преобразованием ***методом булева признака***. Условие выхода из цикла заменяется конъюнкцией из внешнего и внутреннего условий выхода. После завершения такого цикла проверяется по какой причине произошел выход из него и выполняются действия, заданные в преобразуемом (неструктурном) алгоритме.

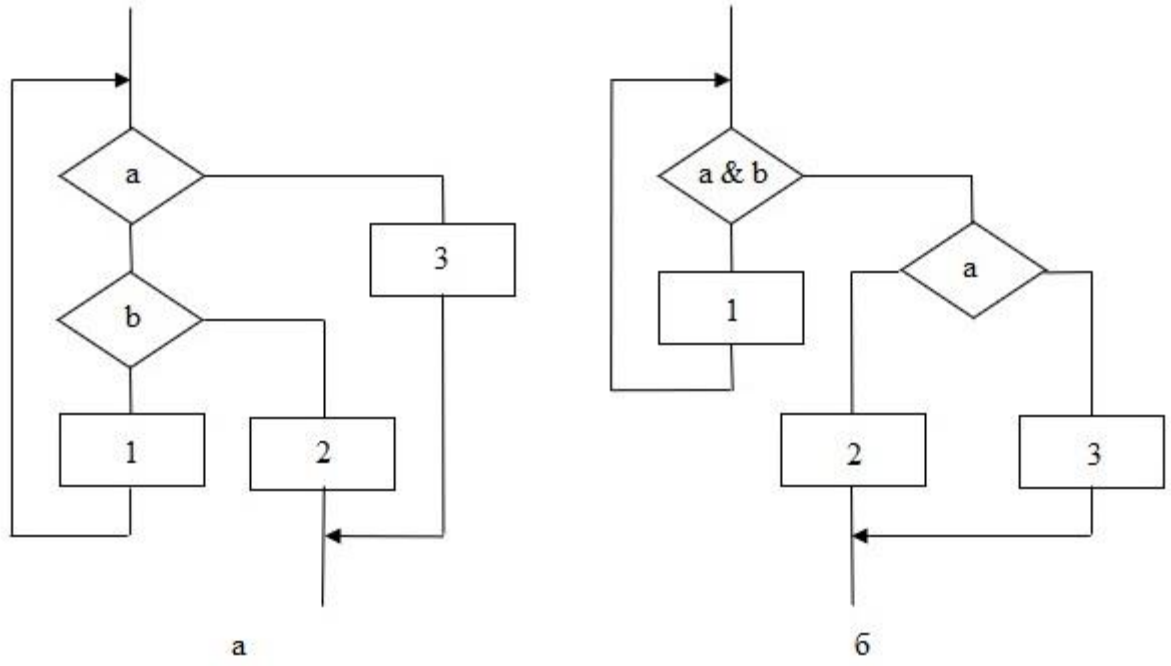


Рис. 1.8. Пример преобразования алгоритма к структурному виду методом булева признака:. а – блок-схема неструктурного алгоритма; б – блок-схема эквивалентного структурного алгоритма

Поскольку метод булева признака не требует введения дополнительных переменных, он является более предпочтительным в указанных выше условиях.

Дополнительная логическая переменная вводится так же и для алгоритмов, содержащих ***циклы с двумя входами***. Такой алгоритм представлен на рис.1.9.

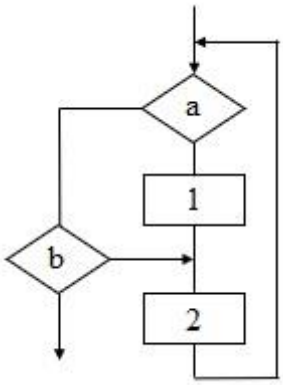


Рис.1.9. Алгоритм с входом внутрь цикла.

Структура программы такого алгоритма с goto, например, может быть такой:

```
m1: if (a)
      if (b) goto m2 else {2; goto m1}
   else {1; 2; goto m1}
m2: end
```

Преобразованная к структурному виду блок-схема этого алгоритма представлена на рис. 1,10.

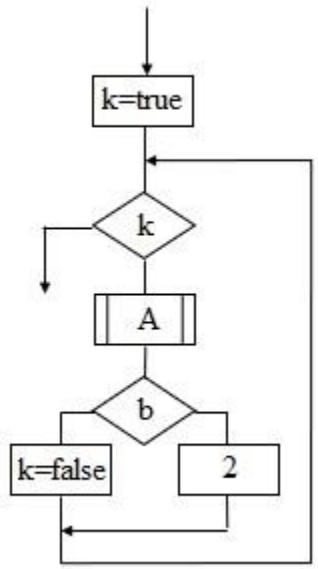


Рис. 1,10. Структурный алгоритм с «преопределенным процессом» A - while(not(a)) {1; 2}

На первой итерации этого алгоритма при a = false будет выполняться A после чего будет a = true, и вычисления продолжатся по левой ветви алгоритма (рис. 1.9). Если же на первой итерации окажется что a = true, то A не выполнится ни разу. Таким образом, нет необходимости использовать if(a) {…} else {…}.

И еще. Очевидно, что алгоритм на рис. 1.9 итерационный и заканчивает свою работу при (a & b) = true, поэтому в качестве условия его выполнения можно было бы написать while(not(a & b)), однако значение «b» к началу работы алгоритма может попросту отсутствовать. Именно поэтому и введена дополнительная переменная k.

Если цикл имеет ***несколько входов***, то для каждого вхождения следует указать действия, которые должны выполниться перед тем, как начать цикл сначала.

1.1.4. Преобразование циклов

Циклы с постусловием (do-while) преобразуются в циклы с предусловием (while) так, как это представлено на рис. 1.11. Для BPEL такое преобразование обязательно.

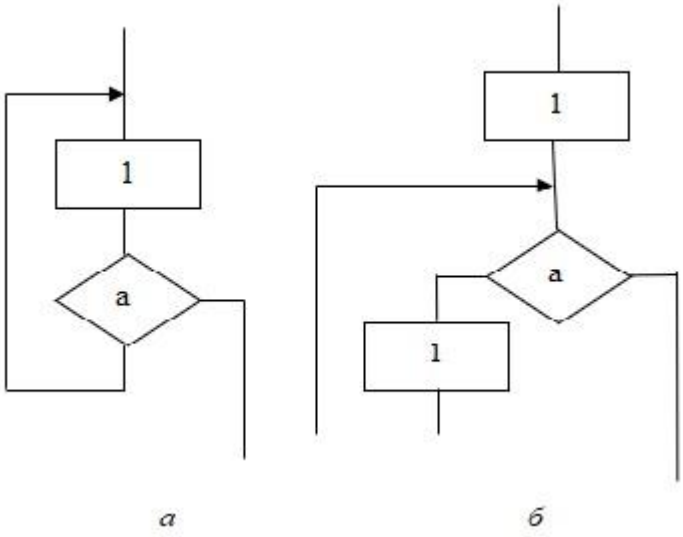


Рис. 1.11. Эквивалентные циклы: а – цикл с постусловием; б – цикл с предусловием

Обратное преобразование циклов в общем случае, естественно, невозможно.

Циклы с параметром for i=n1,n2,h преобразуются в циклы с предусловием (while) так, как это представлено на рис. 1.12.

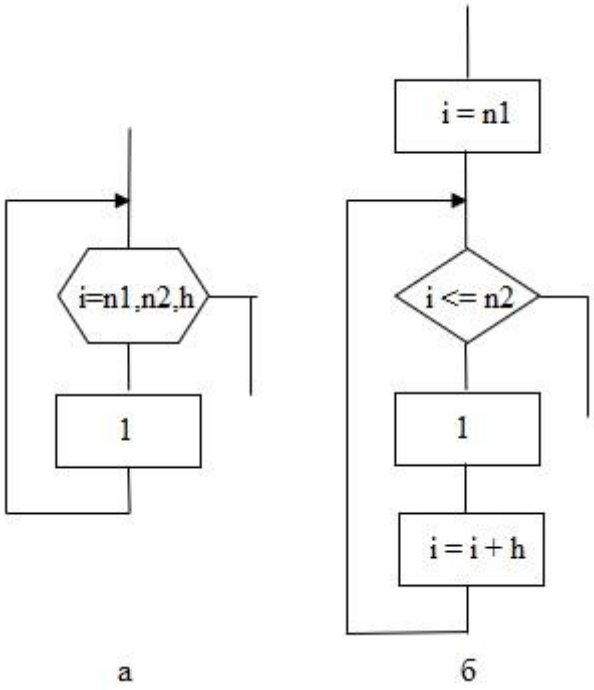


Рис. 1.12. Эквивалентные циклы: а – цикл с параметром; б – цикл с предусловием

Оба цикла эквивалентны, поскольку тело этих циклов («1») выполнится одинаковое число раз при заданных n1, n2 и h.

1.2. Блок-схемы и структуры программ

Алгоритму, задаваемому блок-схемой, всегда соответствует структура программы, реализующая этот алгоритм, и наоборот. Таким образом, эти 2 представления структур эквивалентны и отличаются только используемой нотацией. Неструктурный алгоритм на блок-схеме содержит структуры, у которых более одного входа и (или) одного выхода. Структура программы, реализующая такой алгоритм, неизбежно содержит оператор (операторы) передачи управления (такие, как goto, *break, continue*), и наоборот.

Например, в неструктурном алгоритме (рис. 1.8,а) из цикла по «а» имеется 2 выхода, поэтому структура соответствующей программы будет содержать оператор передачи управления (goto):

```
/* Prog 1
m: if (a)
      if (b) {1; goto m};
      else 2;
   else 3;
```

После преобразования этого алгоритма к структурному виду (рис. 1.8,б):

```
/* Prog 2
while (a && b) 1;
if (a) 2;
else 3;
```

1.3. Эквивалентность алгоритмов

Достаточным условием реализации алгоритма BPMN на языке BPEL является его структурный вид, а необходимым эквивалентность исходного и преобразованного алгоритмов. Эквивалентными (в данном случае) являются алгоритмы тогда и только тогда, когда в них совпадают последовательности действий при одинаковых значениях управляющих переменных. Процесс анализа таких последовательностей называется тестированием «белого ящика» [12]. Такой тип тестирования гарантирует проверку всех независимых маршрутов, поскольку проходятся все ветви *true* и *false*, ветви множественного выбора (ветви оператора switch) и выполняются все циклы (в пределах их границ и диапазонов) [12].

В таблице 1.1 представлены все возможные варианты значений условий и соответствующие им состояния (состояние обозначает выполняющееся действие). Фигурными скобками «{» и «}» выделены повторяющиеся состояния.

Таблица 1.1

Протокол тестирования алгоритмов и структур программ

a	b	Рис. 1.7,а	Рис. 1.7,б	Prog 1	Prog 2
False	False	3	3	3	3
False	True	3	3	3	3
True	False	2	2	2	2
True	True	{1}	{1}	{1}	{1}

Поскольку при заданных значениях условий последовательности действий алгоритмов, представленных на рис. 1.8,а и рис. 1.8,б, и структур программ Prog 1 и Prog 2, реализующих эти алгоритмы, совпадают, они эквивалентны.

1.4. Схемы Дамке

На схемах Дамке символы управляющих структур вложенных блоков располагаются справа от структур блоков, в которые они входят [13]. Последовательность действий, составляющих блок, выполняется сверху вниз (рис. 1.13, 1,14).

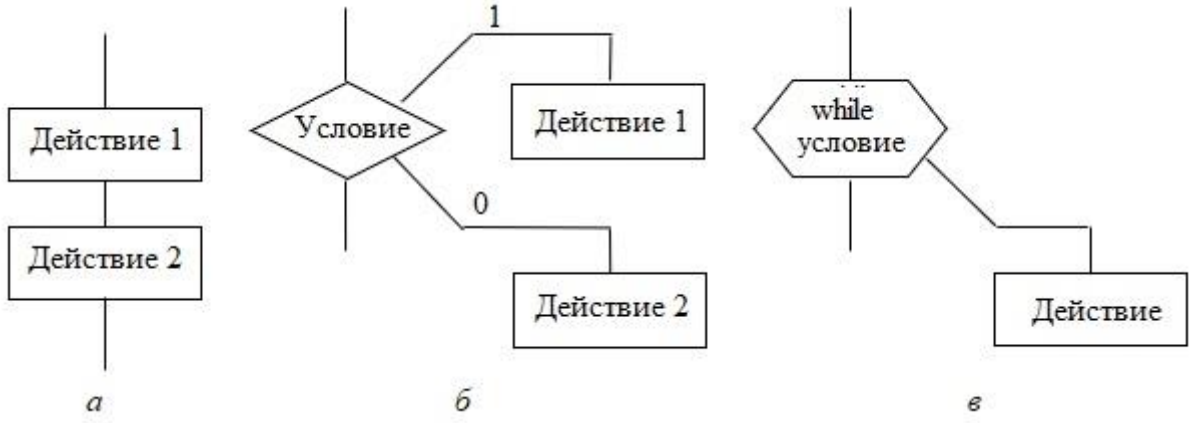


*Рис. 1.13***.** Нотация Дамке, основные структуры:
а – последовательность; б – выбор (if-then-else); в – цикл с предусловием (while-do)

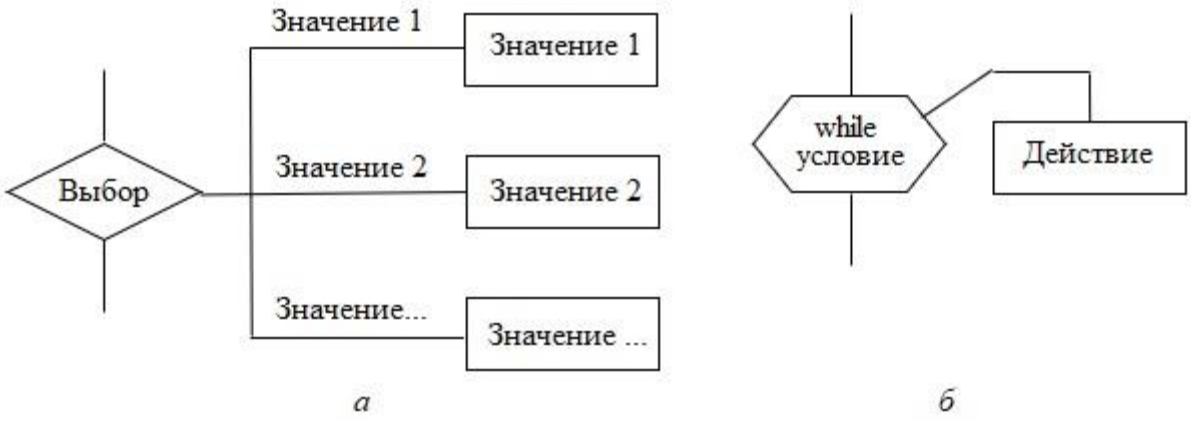


Рис. 1.14. Нотация Дамке, дополнительные структуры:
а – выбор (switch); б – цикл с постуловием (do-while)

Схема Дамке алгоритма рис. 1.8б представлена на рис. 15.

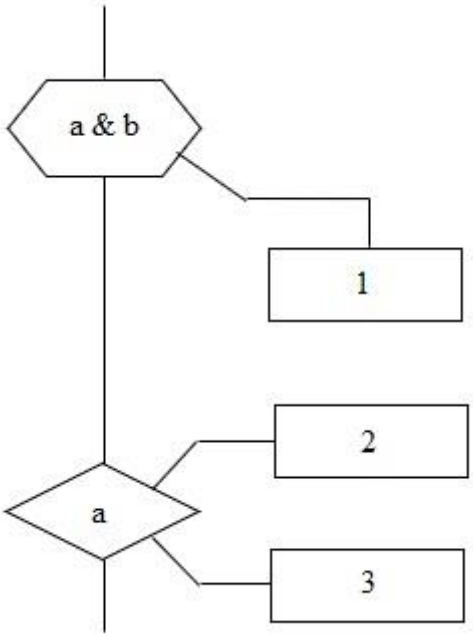


Рис. 1.15. Пример диаграммы Дамке

Алгоритм состоит из последовательности структур цикла и выбора, в первую из которых вложена структура «1», во вторую – структуры «2» и «3».

1.5. Схемы Насси–Шнейдермана

Схемы Насси–Шнейдермана, как и схемы Дамке, предназначены для разработки и документирования структурированных программ. Схемами Насси–Шнейдермана [14] (рис. 1.16 и 1.17) могут быть представлены только структурные алгоритмы. Программа в этой нотации представляется последовательностью вложенных блоков, при этом в отличие от схем Дамке, символы вло-

женных структур располагаются внутри символов тех структур, куда они входят. Таким образом, символы схем Насси–Шнейдермана отражают вложенность тегов, содержащих управляющие операторы языка BPEL.

Рис. 1.16. Нотация Насси–Шнейдермана, основные структуры:
а – последовательность; б –выбор (if-then-else); в – цикл с предусловием (while-do)

Рис. 1.17.. Нотация Насси–Шнейдермана, дополнительные структуры:
а – switch; *б* – цикл с постусловием (do-while)

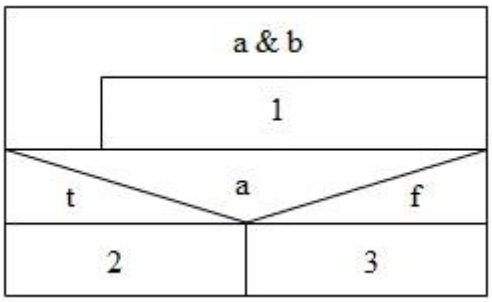


Рис. 1.18. Пример диаграммы Насси–Шнейдермана

На рис. 1.18 представлена диаграмма схема Насси–Шнейдермана алгоритма на рис. 1.8б. Последовательно выполняются цикл с предусловием по a & b, затем в зависимости от значения «a» выбор соответствующего действия.

Такую же структуру, естественно, будет иметь описание БП на языке BPEL поскольку оно будет содержать во вложенных тегах алгоритм, преобразованный к структурному виду.

1.6. Нотация PBMN Aris

BPMN содержит следующие символы для представления БП [15]:

- 9 символов соединяющих элементов (стрелок), которые обозначают потоки операций, данных и ассоциаций;

- обозначения исполнителей и зон ответственности в виде пулов и дорожек;

- 12 символов событий для обозначения их содержания с дополнительным указателем на время наступления события;

- 8 символов задач (действий) с обозначениями их содержания;

- 8 символов подпроцессов с обозначениями их типов;

- 8 символов шлюзов, обозначающих управление слиянием и разделением потоков;

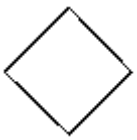
- 3 символа для обозначения объектов данных и символ базы данных;

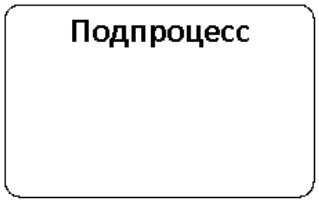
- символы артефактов, обозначающие сообщения, аннотации (комментарии) и группировки БП.

В таблице 1.2 представлены символы BPMN Aris, необходимые для задания алгоритма БП.

Таблица 1.2

Символы алгоритмов БП

Название	Символ	Описание
Поток операций, поток управения (Sequence Flow)		Задает последовательность (до-после) возникновения событий и выполнения действий
Не типизированное стартовое событие (Start event)		Используется для отображения начала процесса
Не типизированное конечное событие (End event)		Используется для отображения окончания процесса
Абстрактная задача		Задача с неопределенным типом
Неопределенный шлюз (Gateway)		При разделении направляет поток в одну исходящую ветвь. При объединении ожидает завершения одной входящей ветви перед инициированием исходящего потока.
Эксклюзивный шлюз (XOR, «Исключающее ИЛИ») Exclusive gateway		Предназначен для разделения потока операций на несколько альтернативный маршрутов, когда в ходе выполнения процесса может быть активирован только один из предложенных маршрутов. Условия пропуска по исходящему

		маршруту задается рядом с соответствующей линией в виде логического выражения
Подпроцесс (Subprocess)		Подпроцесс с неопределенным типом
Символ аннотации		Может использоваться для указания маршрута выхода из шлюза

Алгоритмы БП при анализе и разработке могут содержать другие типы символов событий, задач и шлюзов, представленных в табл. 1.2, за исключением символов, указывающих на параллельность выполнения процессов.

2. ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ

Преобразование блок-схем к структурному виду

Блок-схема неструктурного алгоритма представлена на рис. 2.1 содержит 2 цикла – внутренний цикл с предусловием для «2» и внешний цикл с постусловием по «с». Цикл для «2» не является структурным, поскольку имеет 2 выхода. Преобразуем этот цикл методом булева признака (см. рис. 1.7). Результат преобразования представлен на рис. 2.2.

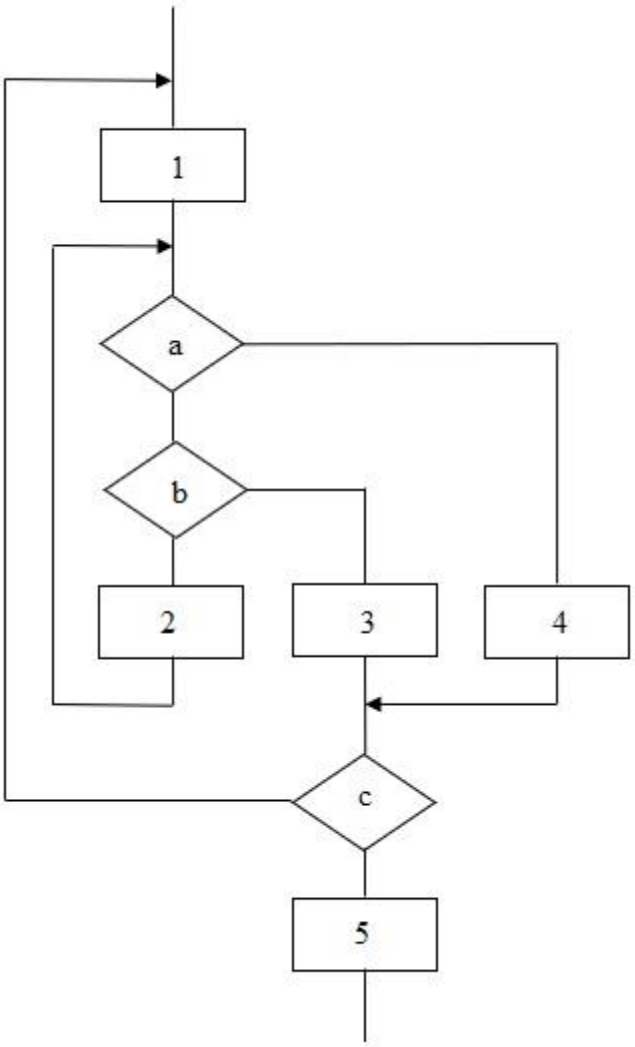


Рис. 2.1. Пример блок-схемы неструктурного алгоритма

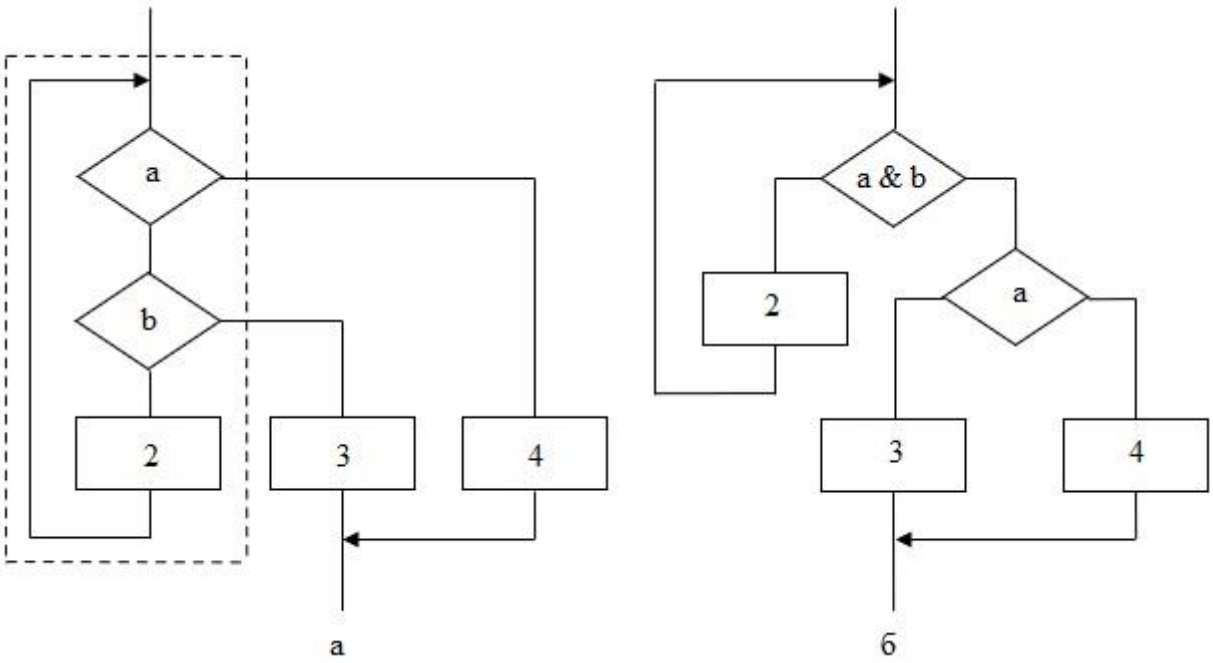


Рис. 2.2. Преобразование цикла неструктурного алгоритма к структурному виду:

а – неструктурный алгоритм; б – эквивалентный структурный алгоритм

Выделенный преобразованный фрагмент исходной блок-схемы обозначим символом «предопределенного процесса» (так на блок-схемах обозначаются, например, процедуры и функции программ) и назовем «А». Подставим «А» в исходную блок-схему. Результат подстановки представлен на рис. 2.3.

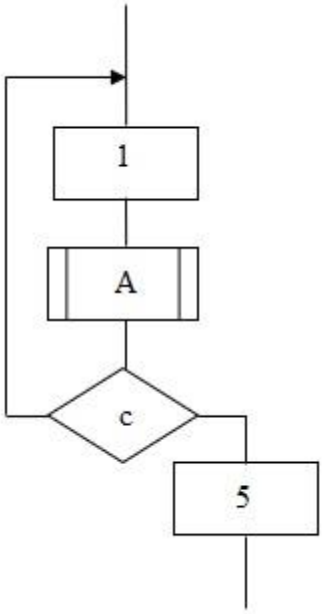


Рис. 2.3. Блок схема структурного алгоритма с «предопределенным процессом»

Преобразуем цикл с постусловием по «с» в цикл с предусловием. Результат представлен на рис. 2.4.

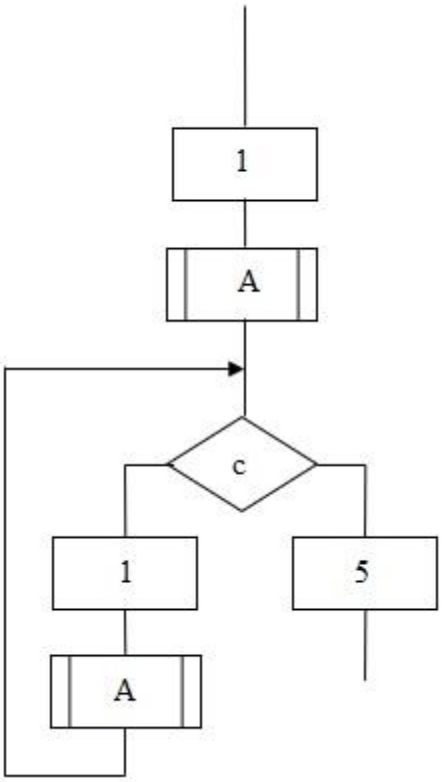


Рис. 2.4. Блок схема структурного алгоритма, не содержащего цикл с постусловием

Наконец, произведем «подстановку» – заменим «А» блок-схемой его алгоритма. Результат такой подстановки представлен на рис. 2.5.

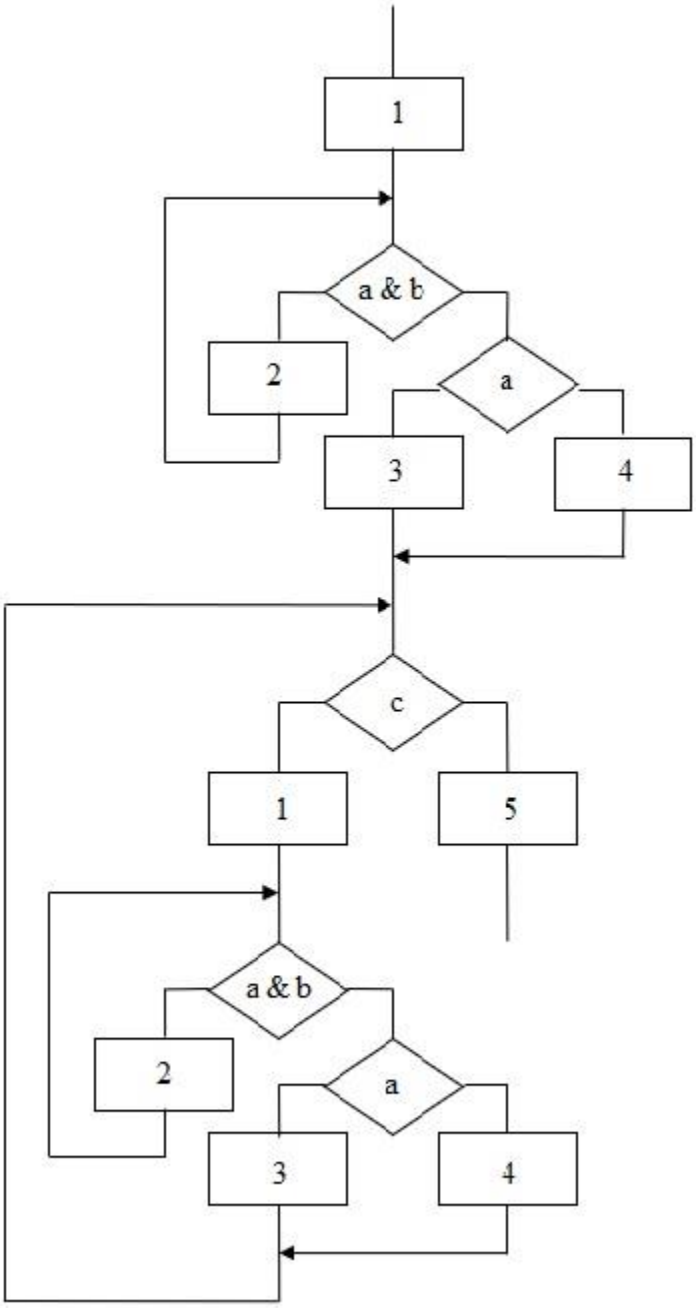


Рис. 2.5. Блок-схема алгоритма (см. рис. 2.1), преобразованного к структурному виду

Структуры программ

Структура программы алгоритма на рис. 2.1 может быть представлена в синтаксисе C (и) следующим образом:

```
/* bad 1
main()
{
   do 1;
   while (a)
      if (b) 2;
      else {3; goto m}
   4;
   m: while (c);
   5
}
```

Или в стиле MODULA и Python, где оператор goto отсутствует по определению:

```
/* bad 2
main()
{
   do {1;
        if (not a) {4; continue};
        if (not b) {3; continue};
       2}
   while (c);
   5
}
```

Структура программы эквивалентного структурного алгоритма (рис. 2.5) имеет вид:

```
main()
{
   1;
   while (a && b) 2;
   if (a) 3;
   else 4;
   while (c) {1;
      while ((a && b) 2;
      if (a) 3;
      else 4;}
   5
}
```

Структура программы содержит только структурные операторы и не содержит операторы передачи управления, овпоэтому такой алгоритм БП может быть реализован в BPEL.

Эквивалентность алгоритмов и структур программ

Проверим эквивалентность исходного и преобразованного алгоритмов, а также соответствующих им структур программ. Результат проверки представлен в таблице 2.1.

Таблица 2.1

Последовательности действий при заданных значениях управляющих переменных

a	b	c	Неструктурный алгоритм		Структурный алгоритм	
			Блок схема	Структура программы	Блок схема	Структура программы
False	False	false	1,4.5	1,4,5	1,4,5	1,3,5
False	False	true	1,4, {1,4}	1,4, {1,4}	1,4, {1,4}	1,3, {1,3}
False	True	false	1,4,5	1,4,5	1,4,5	1,4,5
False	True	true	1,4, {1,4}	1,4 {1,4]	1,4. {1,4}	1,4 {1,4}
True	False	false	1,3,5	1,3,5	1,3,5	1,3,5
True	False	true	1,3, {1,3}	1,3 {1,3]	1,3, {1,3}	1,3, {1,3]
True	True	false	1,{2}	1,{2}	1, {2}	1,{2}
True	True	true	1,{2}	1,{2}	1, {2}	1, {2}

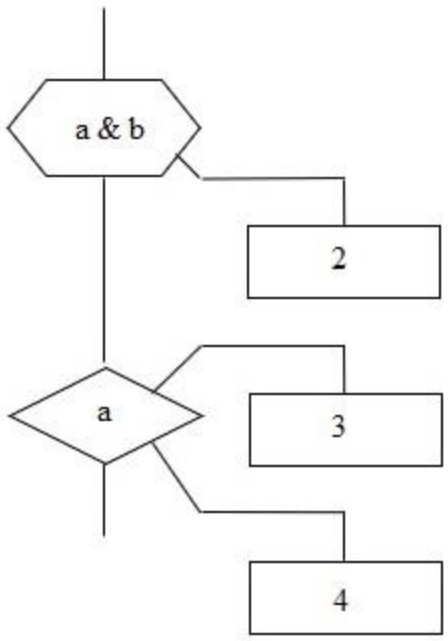
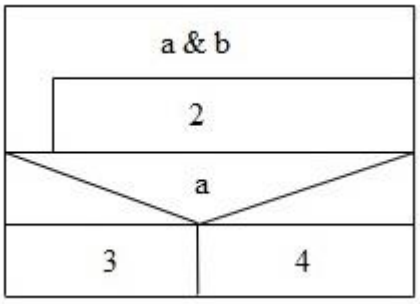
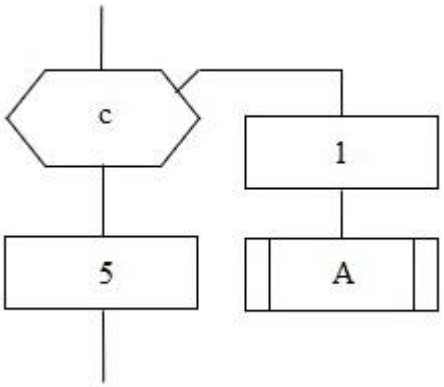
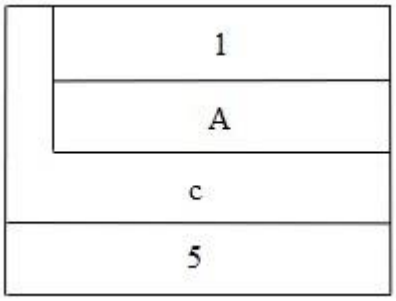
Поскольку последовательности действий, заданные блок-схемами на рис. 2.1 и 2.5, а также последовательности состояний программ исходного и преобразованное алгоритмов совпадают, алгоритмы эквивалентны.

Схемы Дамке и Насси–Шнейдермана

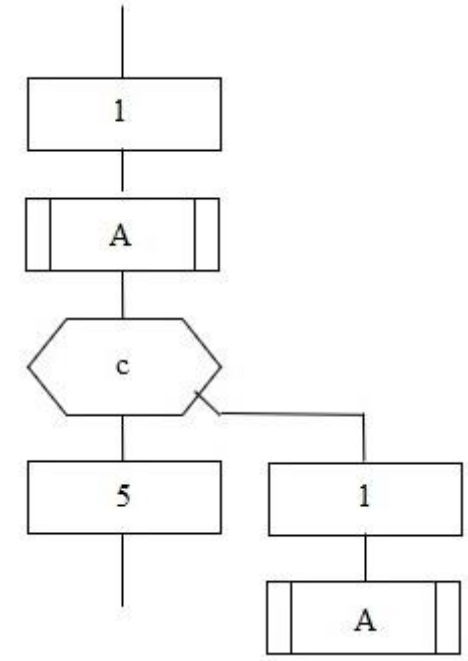
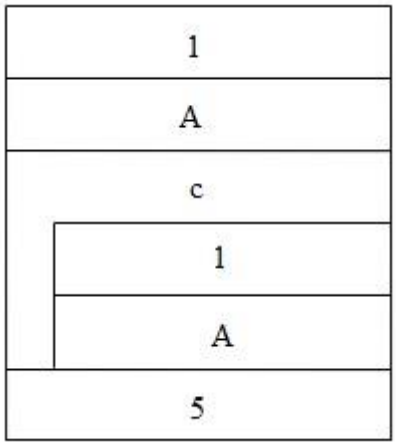
Схемы Дамке и Насси–Шнейдермана последовательности преобразований блок-схемы исходного алгоритма (рис. 2.2 – 2.5) представлены в таблице 2.2.

Таблица 2.2.

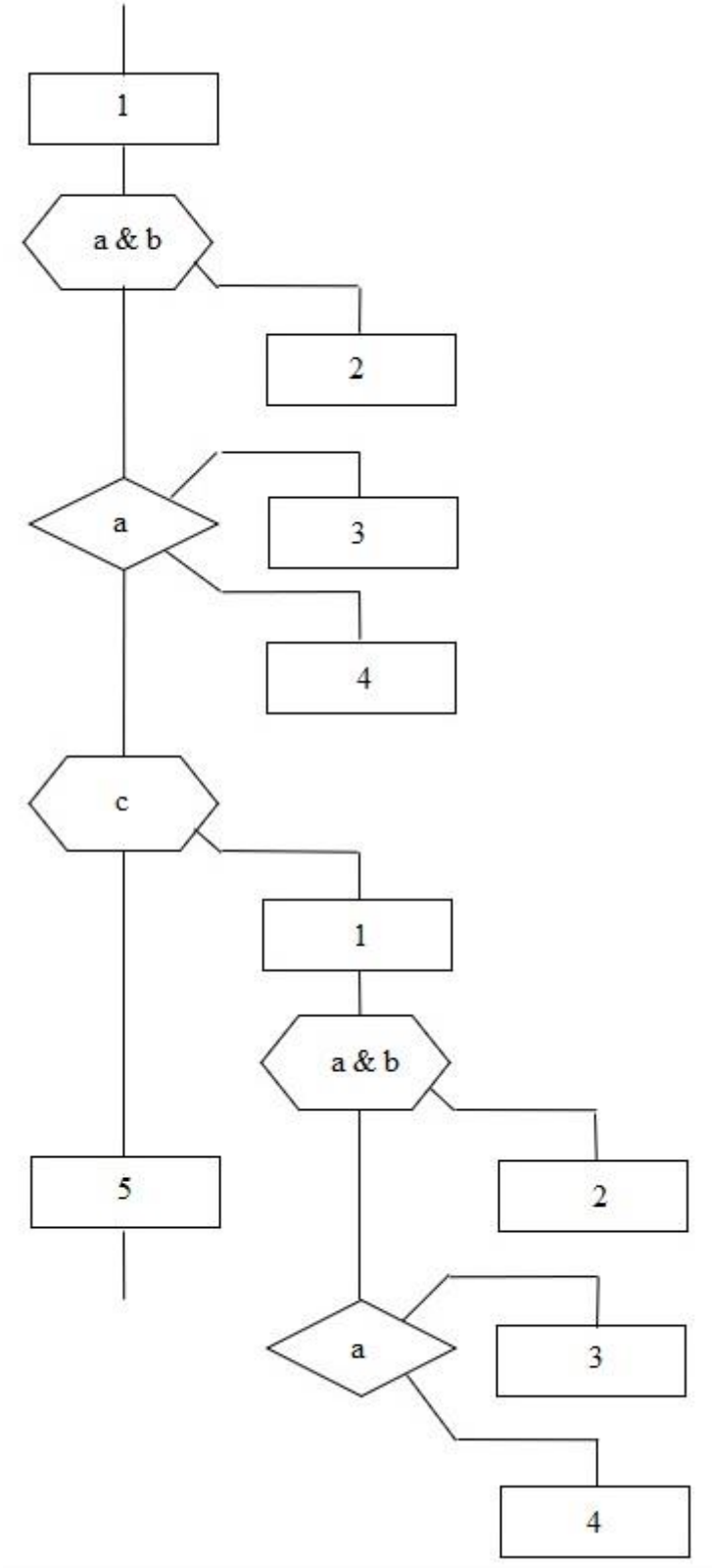
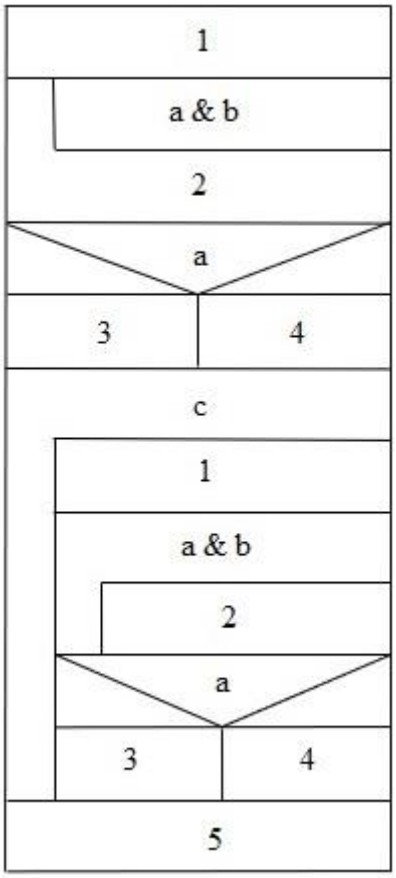
Последовательность преобразований блок-схемы алгоритма в нотации Дамке и Насси–Шнейдермана

Схемы Дамке	Схемы Насси–Шнейдермана
	
	

Продолжение табл. 2.2

Схемы Дамке	Схемы Насси–Шнейдермана
	

Окончание табл. 2.2

Схемы Дамке	Схемы Насси–Шнейдермана
	

Диаграммы BPMN

Диаграммы BPMN, представляющие исходный и эквивалентный ему структурный алгоритмы, представлены на Рис. 2.6 и Рис. 2.7.

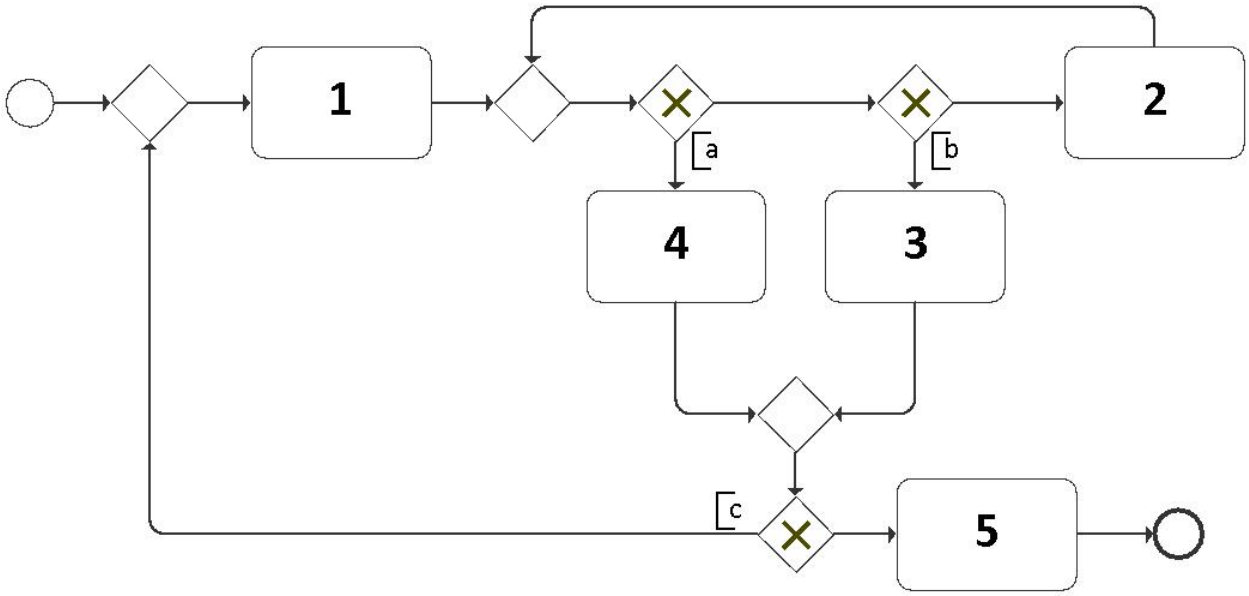


Рис. 2.6. Диаграмма BPMN бизнес-процесса, априорно содержащая ошибки трансляции в BPEL

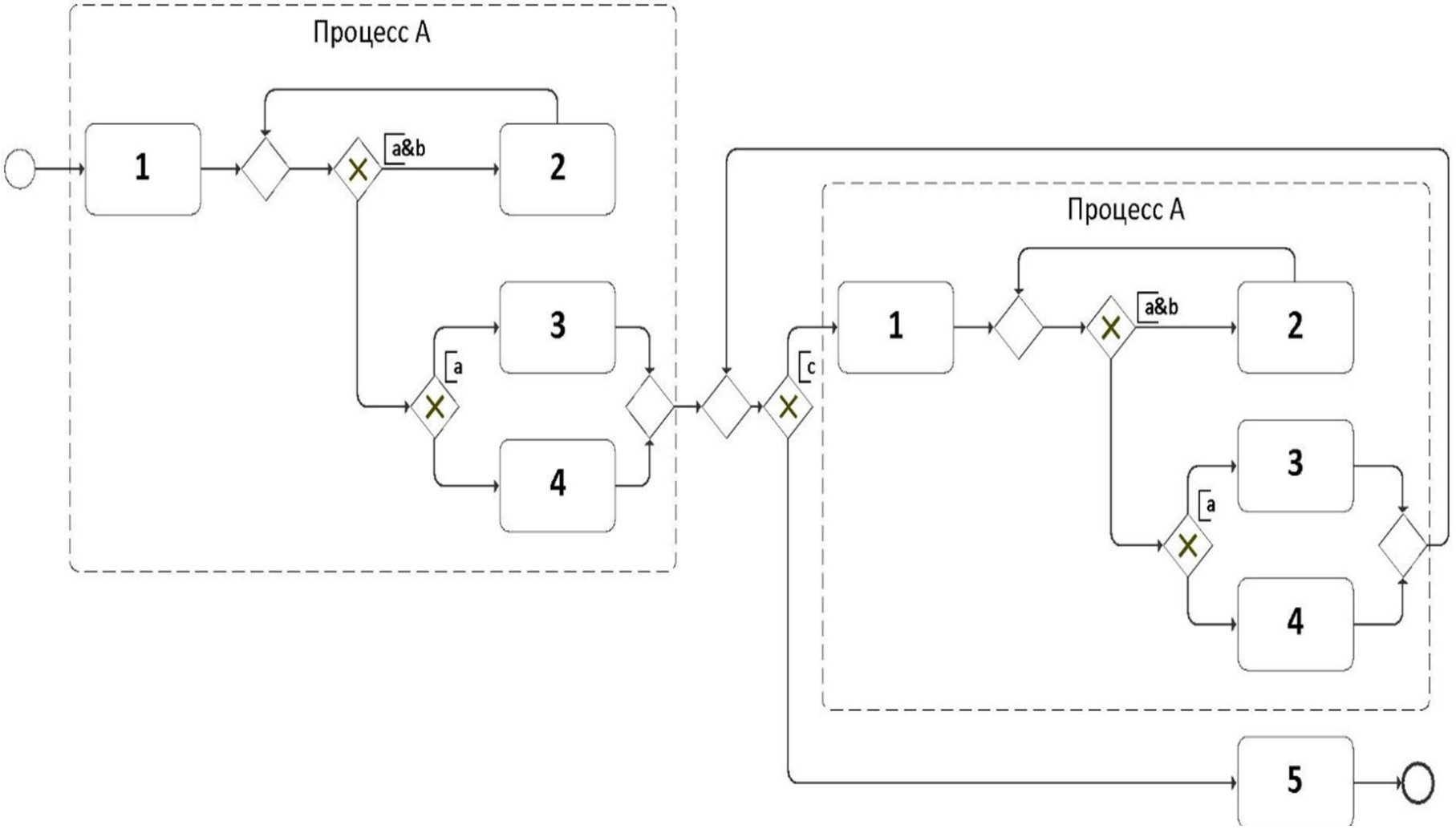


Рис. 2.7. Диаграмма BPMN бизнесс-процесса, допускающего трансляцию в BPEL

СОДЕРЖАНИЕ ОТЧЕТА

1. Последовательность преобразований блок-схемы заданного алгоритма с выделенными и преобразованными неструктурными фрагментами.

2. Синтаксические структуры программ исходного и преобразованного алгоритмов в синтаксисе С (Си).

3. Протокол тестирования эквивалентности блок-схем и соответствующих им синтаксических структур программ.

4. Последовательность схем Дамке структурных алгоритмов, использованных в процессе преобразования.

5. Схема Насси–Шнейдермана преобразованного алгоритма.

6. Диаграммы BPMN заданного и структурного алгоритма БП.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1.Чем отличается оркестровка (orchestration) от хореографии (choreography) web-сервисов в архитектуре SOA?

2. Какими свойствами должен обладать алгоритм бизнес-процесса для реализации его на языке BPEL?

3. Каковы причины появления семантических ошибок в процессе выполнения БП, реализованных на языке PBEL?

4. Каким образом может быть представлена структура выбора switch структурами выбора if-then-else?

5. Достаточно ли базовых алгоритмических структур для представления любого алгоритма?

6. Какова причина дублирования управляющих структур при преобразовании алгоритмов к структурному виду?

7. Какие алгоритмы не могут быть представлены схемами Дамке?

8. Какие алгоритмы могут быть представлены схемами Насси–Шнейдермана?

9. Какие типы потоков могут быть представлены в BPMN?

10. Какие символы событий не имеют входов?

11. Какие типы задач могут быть представлены в BPMN?

12. Чем разтличаются эксклюзивные и неэксклюзивные шлюзы?

СПИСОК ЛИТЕРАТУРЫ

1. Оркестровка и хореография Web-сервисов. URL: https://www.osp.ru/os/2004/11/184785

2. Сервис-ориентированная архитектура (SOA). URL: https://habr.com/ru/company/mailru/blog/342526/

3. Язык выполнения бизнес-процессов – Business Process Execution Language. URL:https://ru.abcdef.wiki/wiki/Business_Process_Execution_Language

4. Артамонов И. В. Оркестровка и хореография // Моделирование бизнес-транзакций. – Иркутск: Издательство Байкальский государственный университет экономики и права, 2016. С. 39 – 45. URL: https://artamonoviv.ru/articles/business_process_orchestration_and_choreography/

5. Бизнес-процессы с использованием диалекта XML BPEL (bourabai.kz). URL: http://bourabai.kz/tpoi/bpel.htm

6. Алгоритмы: Учеб. Пособие.– М.: Мир науки, 2019. Сетевое издание. URL: https://izd-mn.com/PDF/33MNNPU19.pdf

7. BPMN 2.0. Из чего состоит модель бизнес-процесса. URL: https://rzbpm.ru/knowledge/bpmn-2-0-iz-chego-sostoit-model-biznes-processa.html

8. Структурное программирование. UML: http://ermak.cs.nstu.ru/cprog/html/033.htm

9. Схема бизнес-процесса – краткий алгоритм создания (rzbpm.ru). URL: https://rzbpm.ru/knowledge/sozdanie-sxemy-biznes-processa-dlya-neterpelivyx.html

10. Преобразование BPMN в BPEL: почему и как. URL: http://www. interface.ru/home.asp?artId=20860

11. Теорема о структурированной программе Structured program theorem. URL: https://wikithe.ru/wiki/Structured_program_theorem

12. Тестирование программного обеспечения. Базовый курс. URL: https://svyatoslav.biz/software_testing_book/

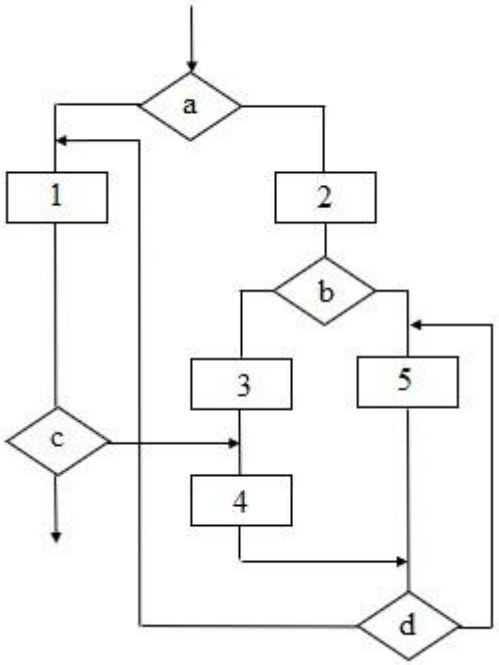
13. Метод Дамке. URL: https://studfile.net/preview/3676187/page:14/

14. Схемы Насси–Шнейдермана. URL: https://studfile.net/preview/3676187/ page:15

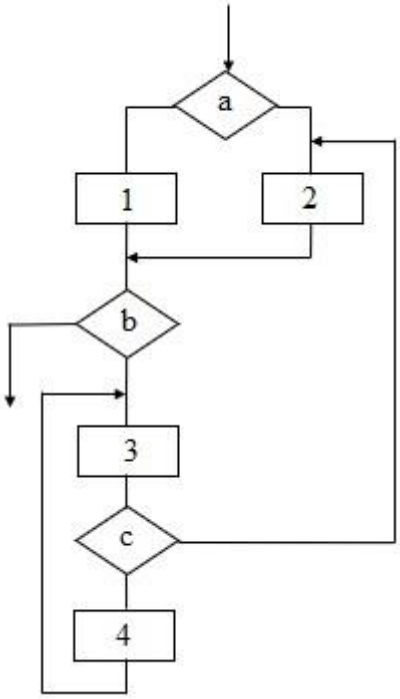
15. Business Process Modeling Notation Specification. Графический язык моделирования бизнес-процессов BPMN. Спецификация (избранные главы). URL: https://plansys.ru/download/BPMN_notation.pdf

ПРИЛОЖЕНИЕ

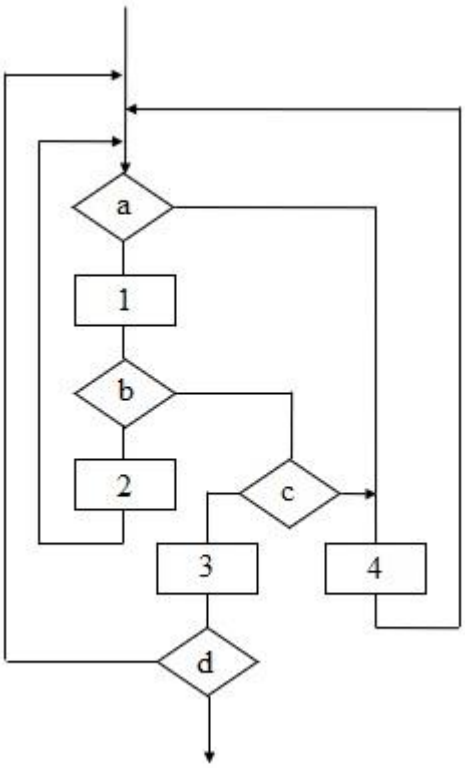
Варианты заданий



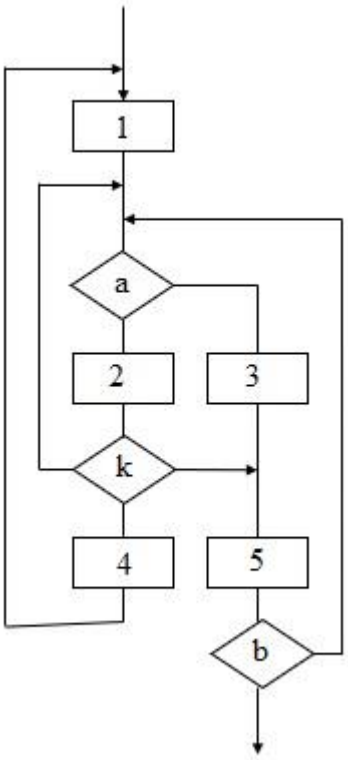
Вариант 1



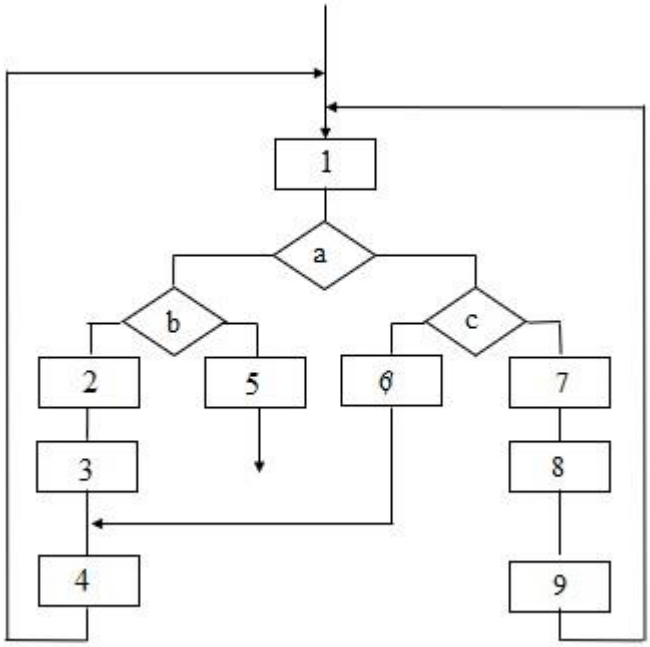
Вариант 2



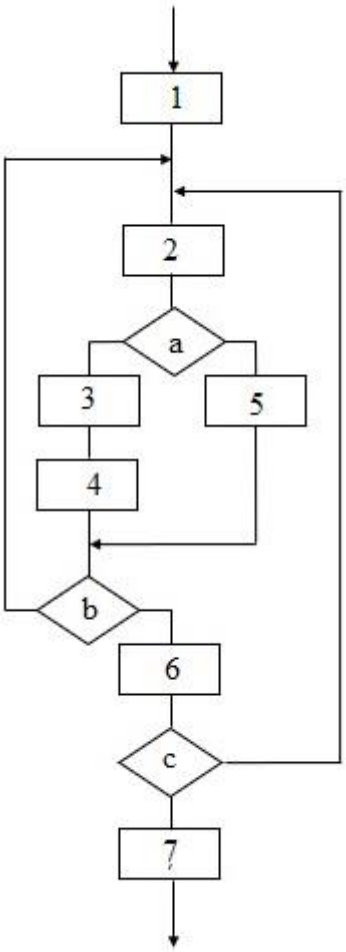
Вариант 3



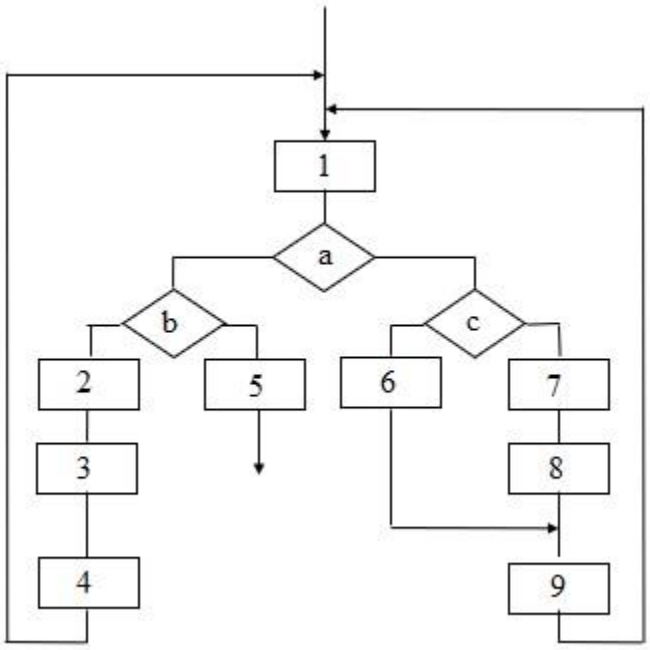
Вариант 4



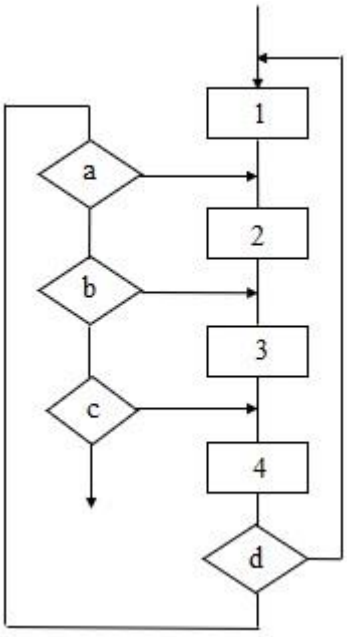
Вариант 5



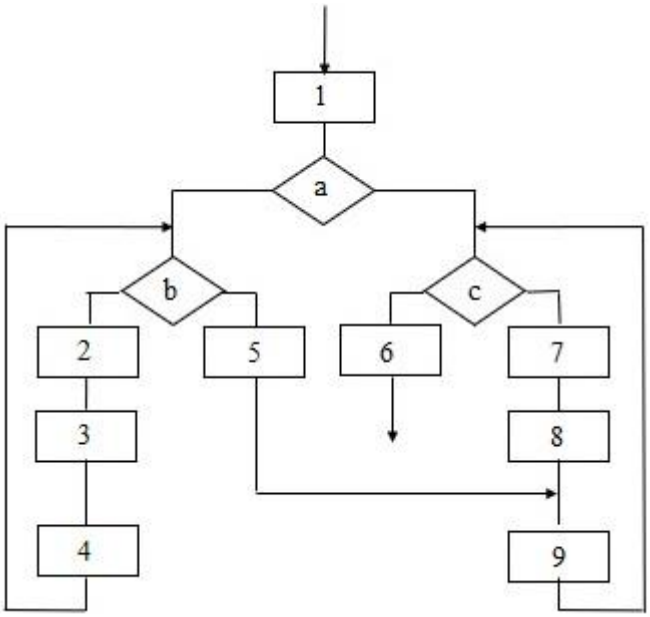
Вариант 6



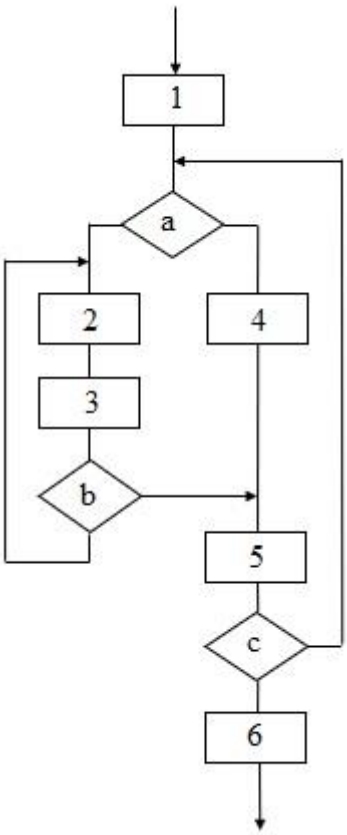
Вариант 7



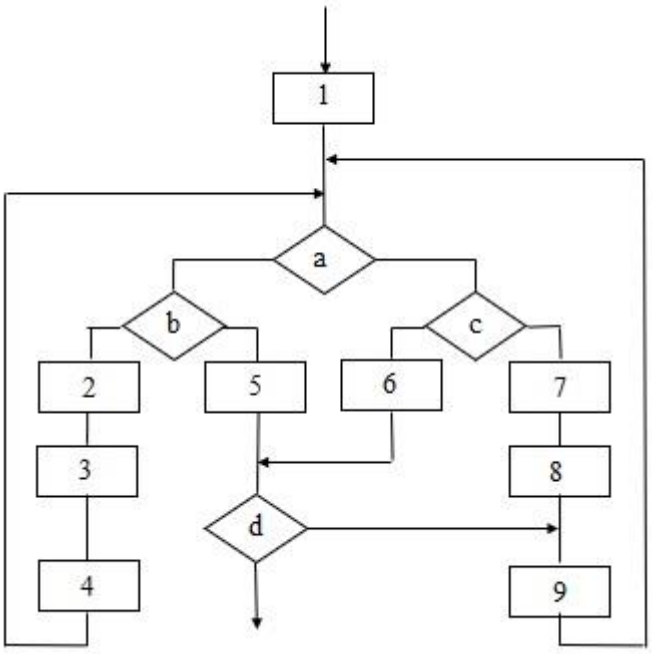
Вариант 8



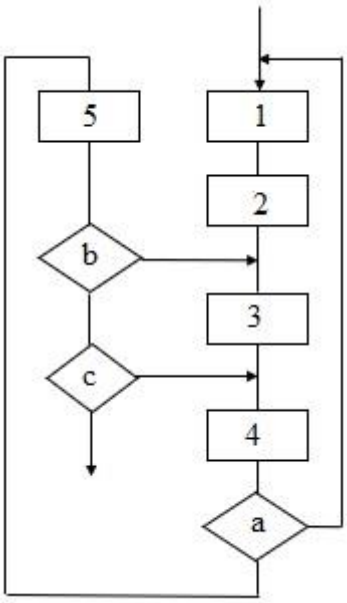
Вариант 9



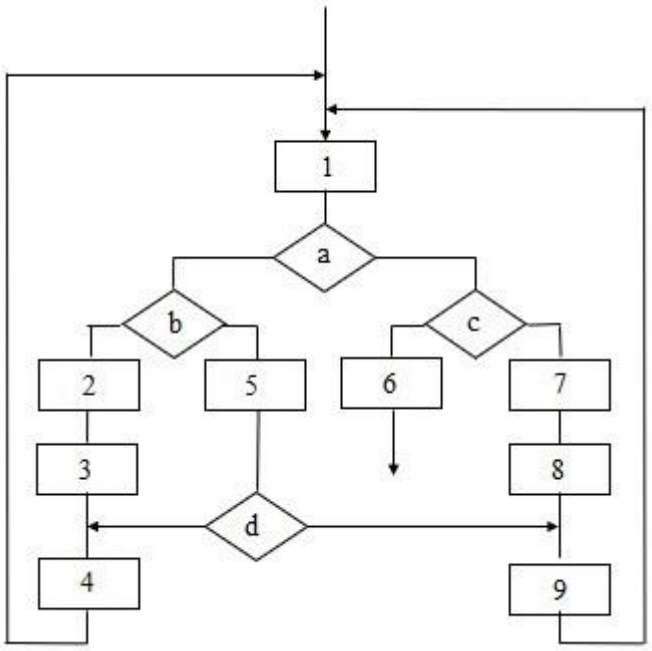
Вариант 10



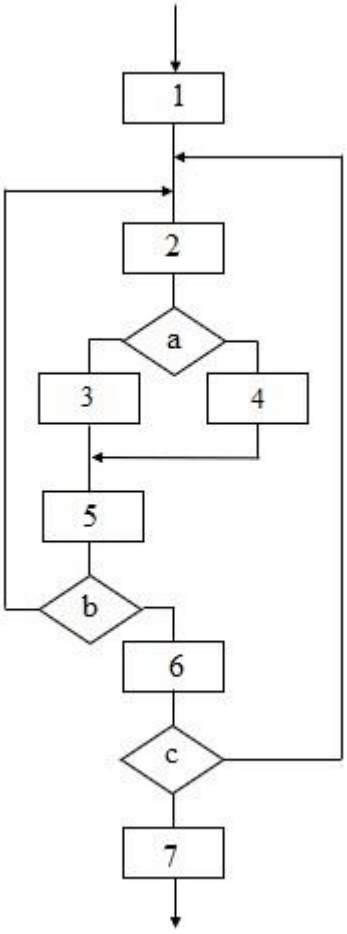
Вариант 11



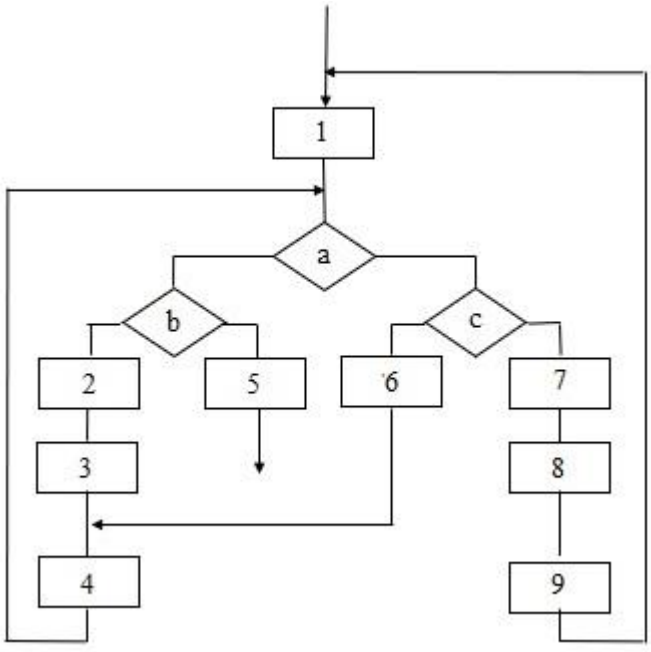
Вариант 12



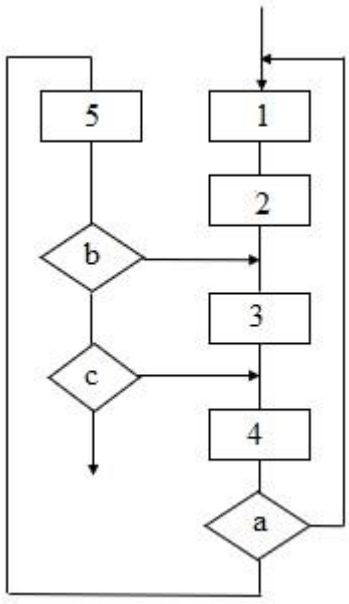
Вариант 13



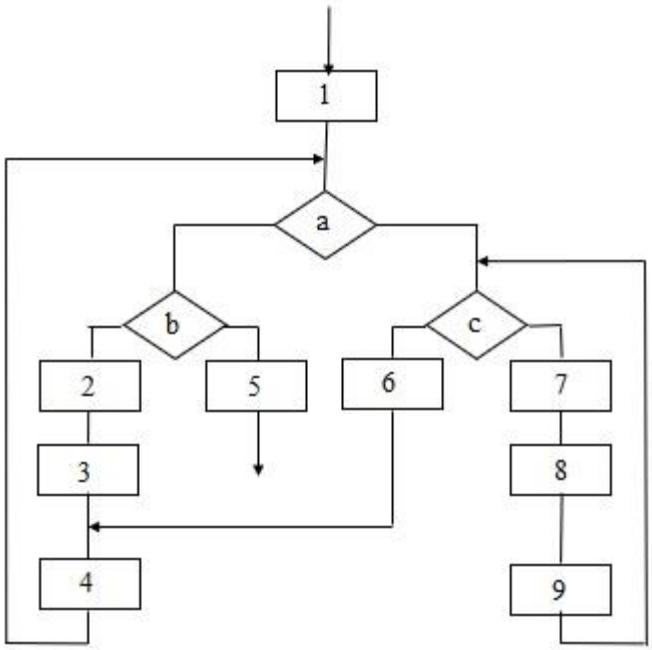
Вариант 14



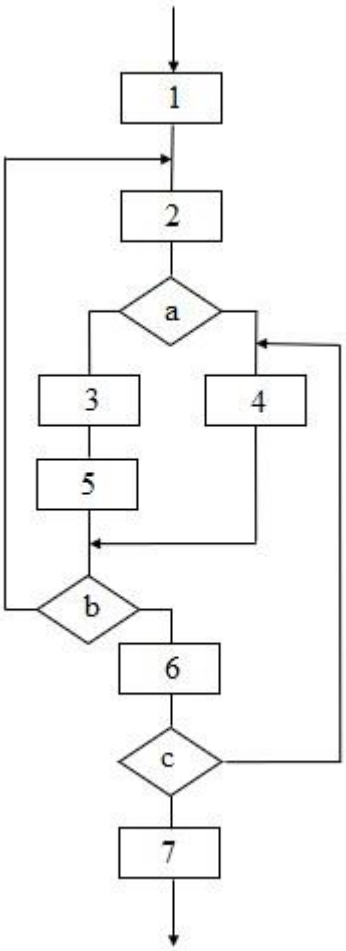
Вариант 15



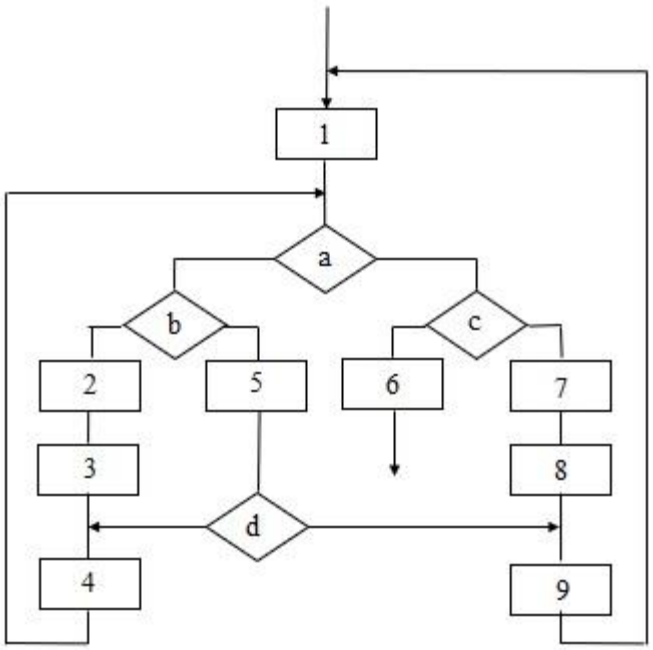
Вариант 16



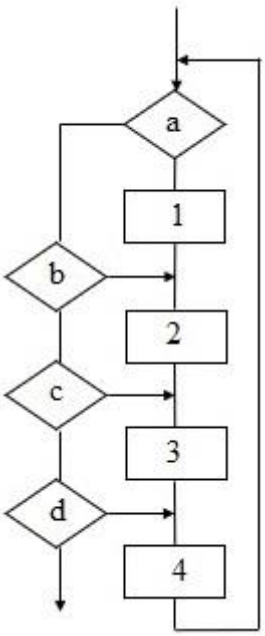
Вариант 17



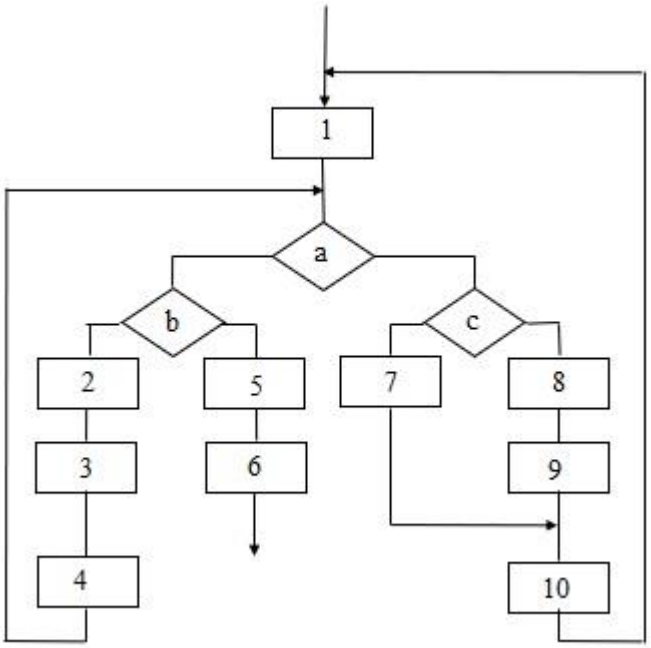
Вариант 18



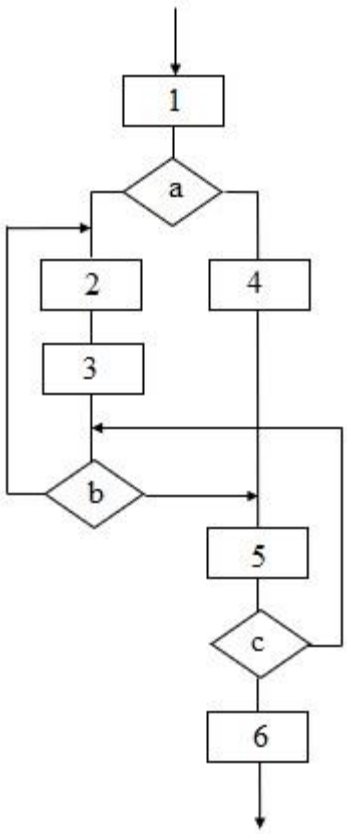
Вариант 19



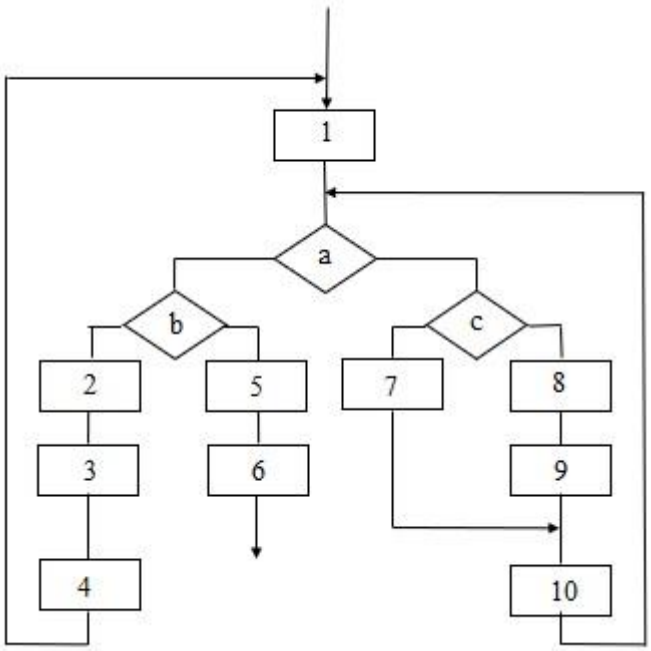
Вариант 20



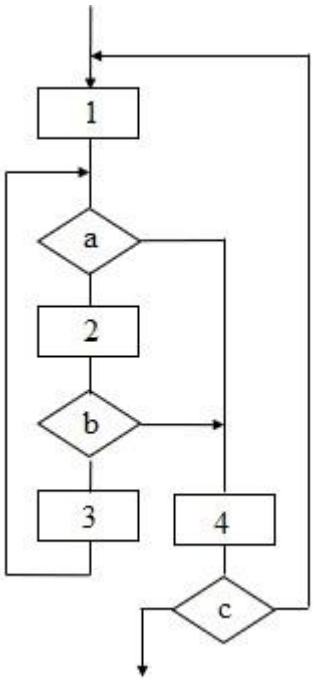
Вариант 21



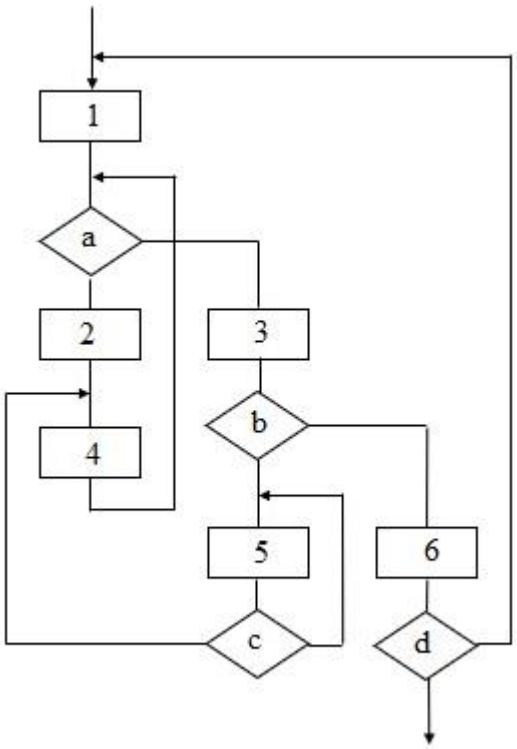
Вариант 22



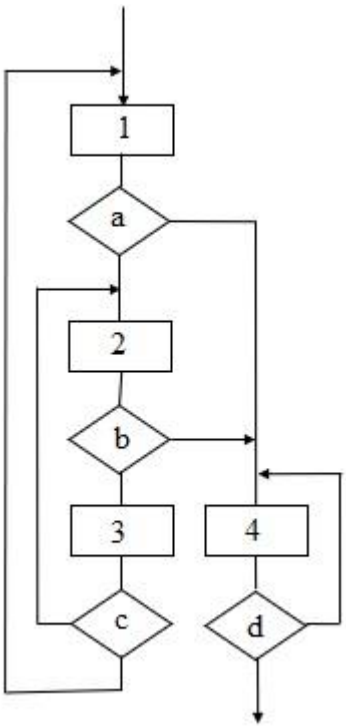
Вариант 23



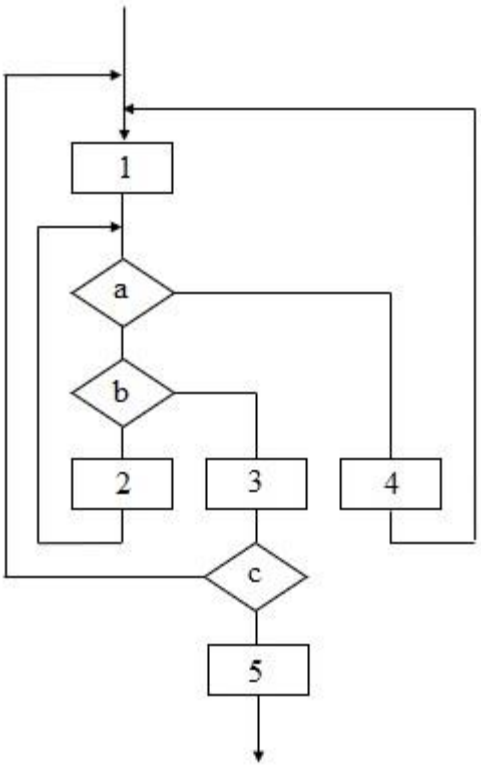
Вариант 24



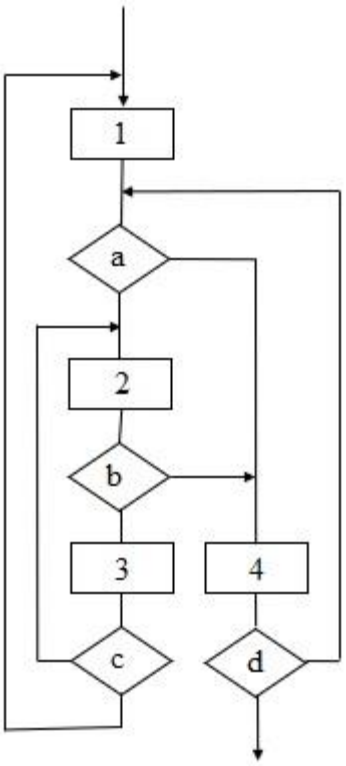
Вариант 25



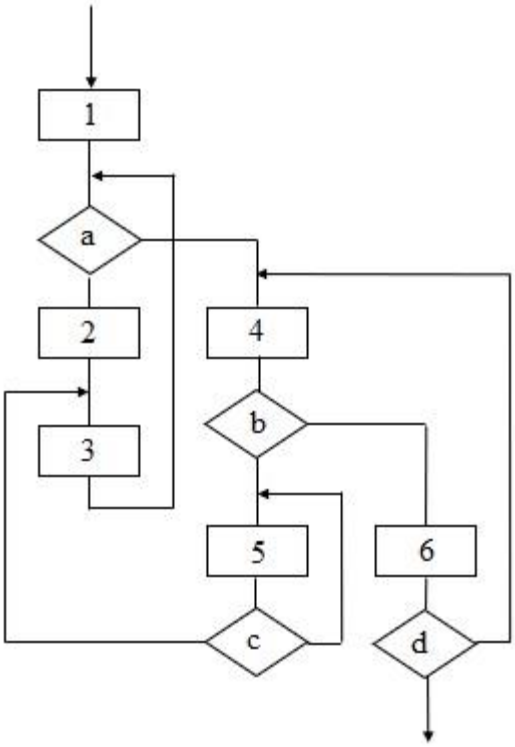
Вариант 26



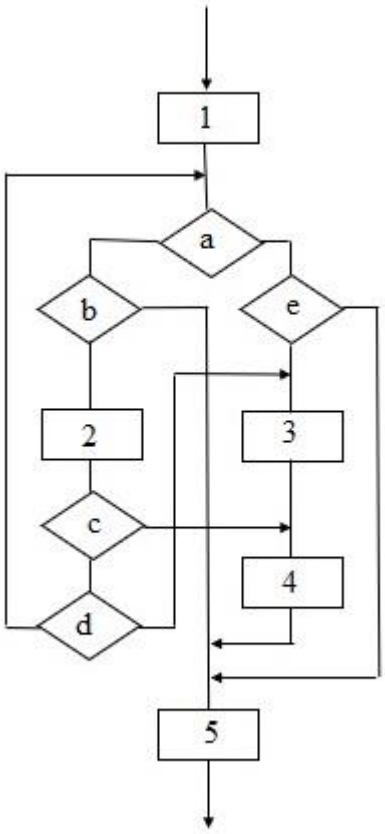
Вариант 27



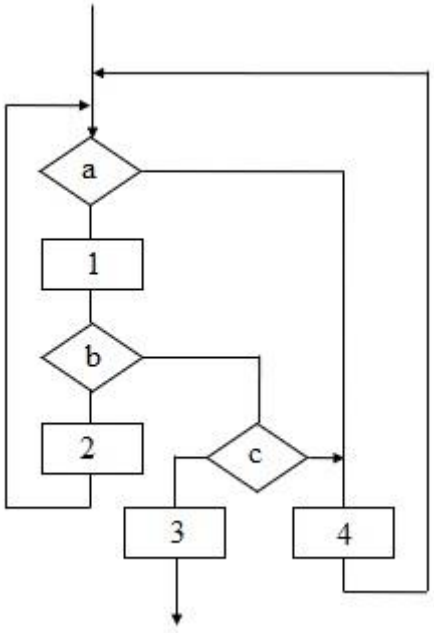
Вариант 28



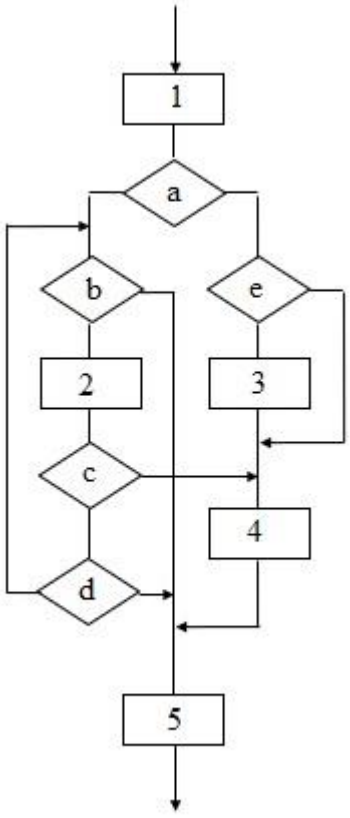
Вариант 29



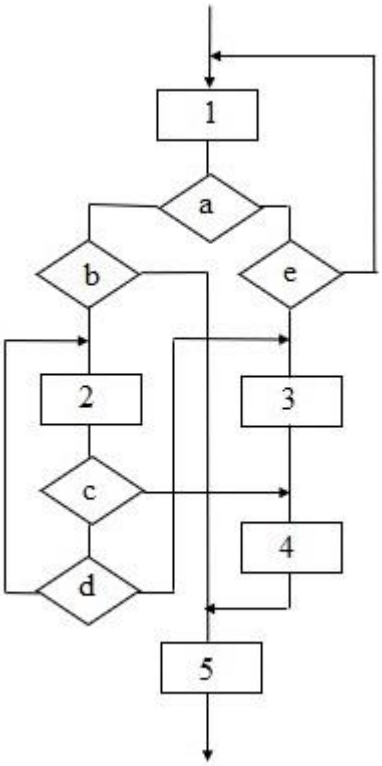
Вариант 30



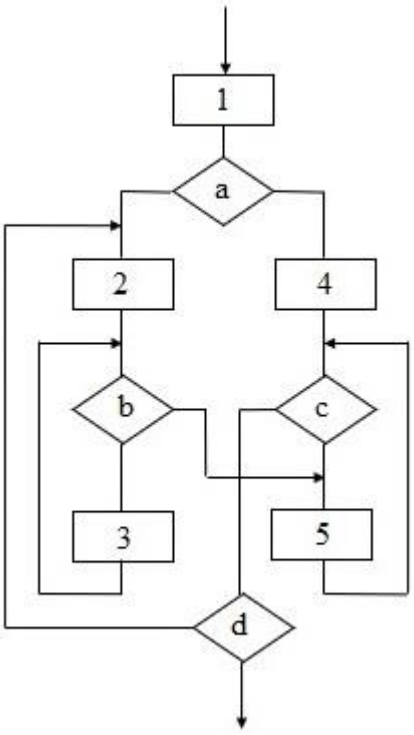
Вариант 31



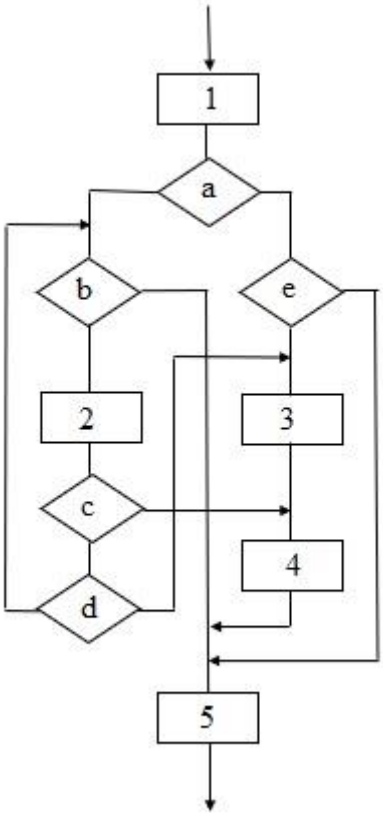
Вариант 32



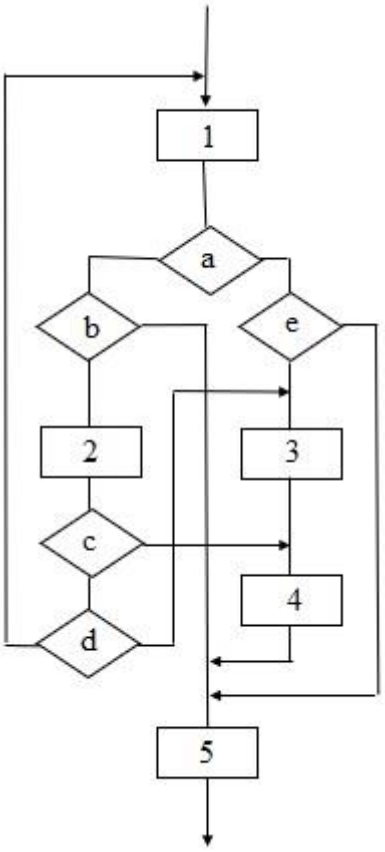
Вариант 33



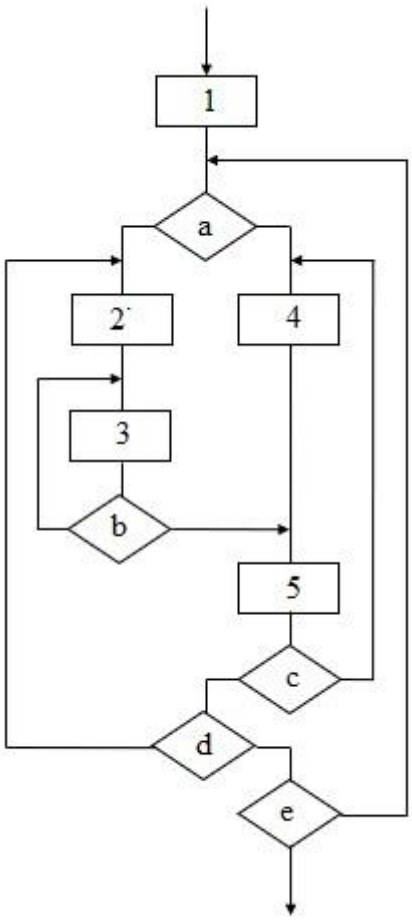
Вариант 34



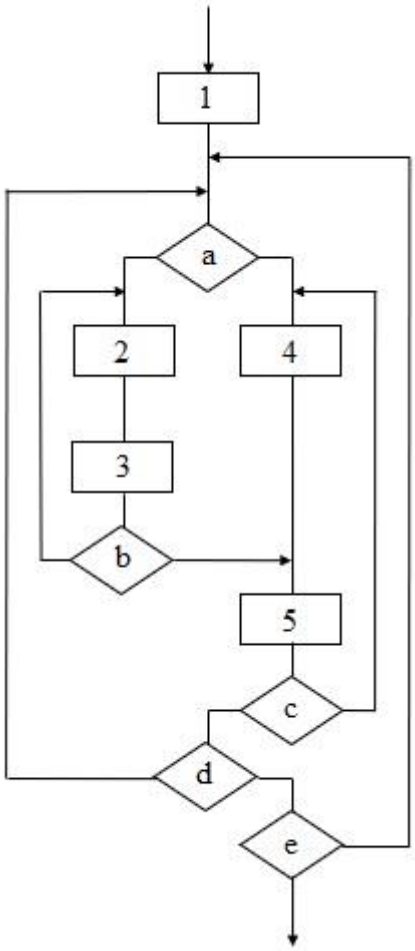
Вариант 35



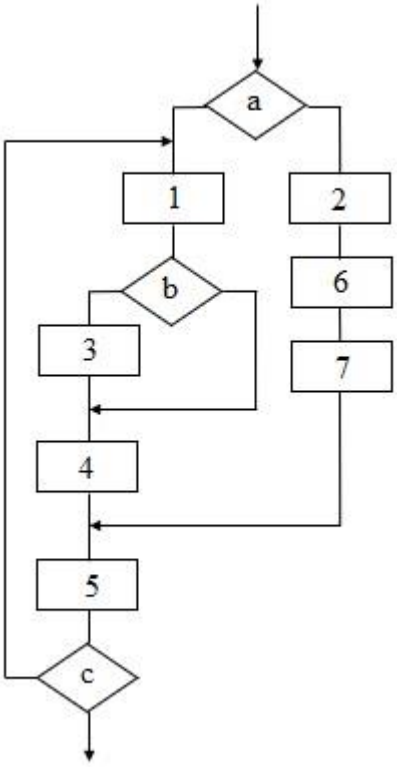
Вариант 36



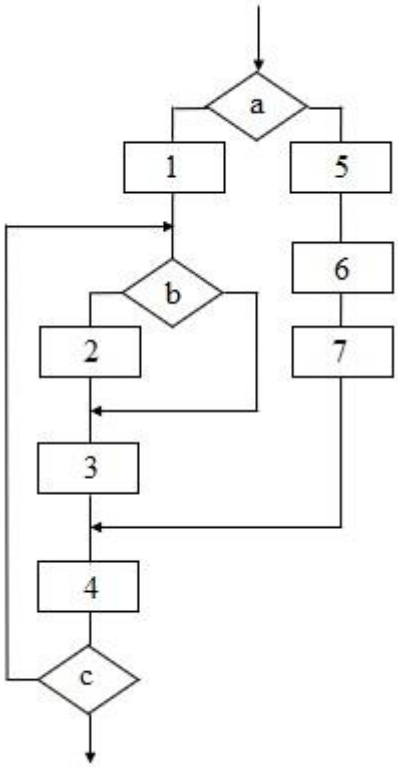
Вариант 37



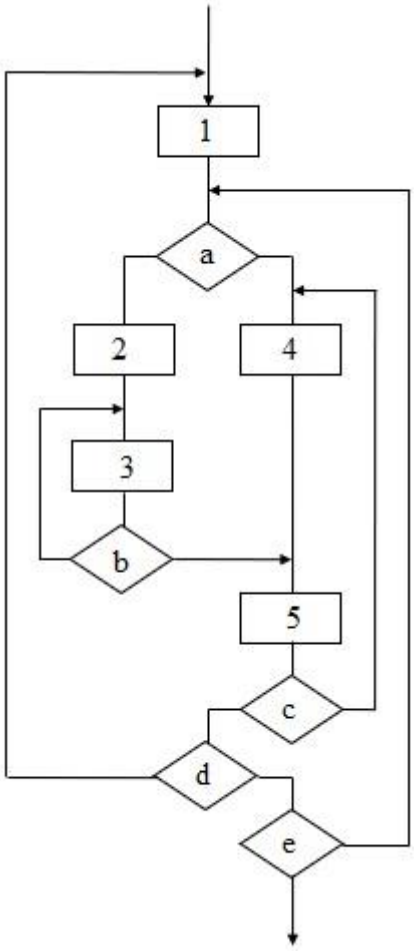
Вариант 38



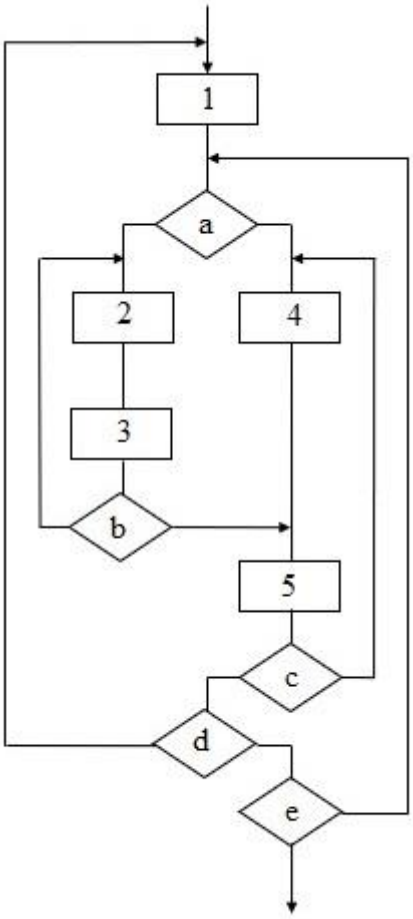
Вариант 39



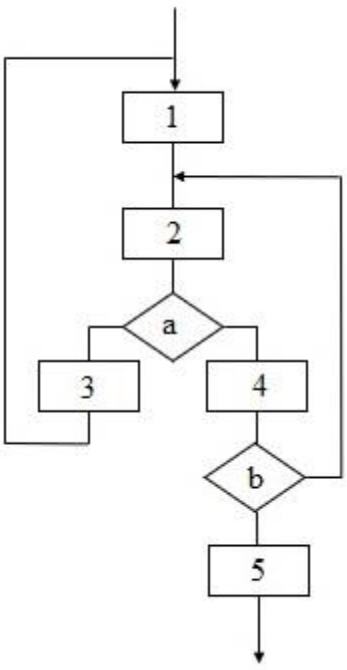
Вариант 40



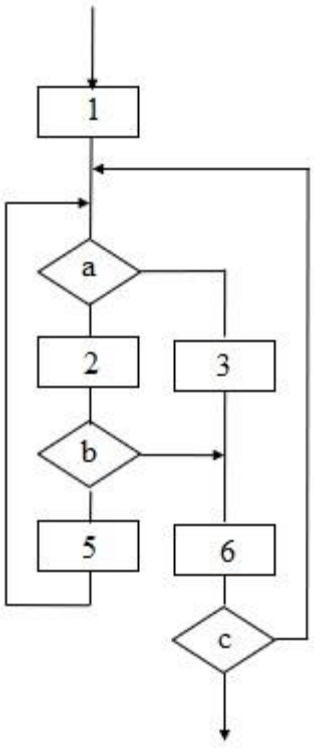
Вариант 41



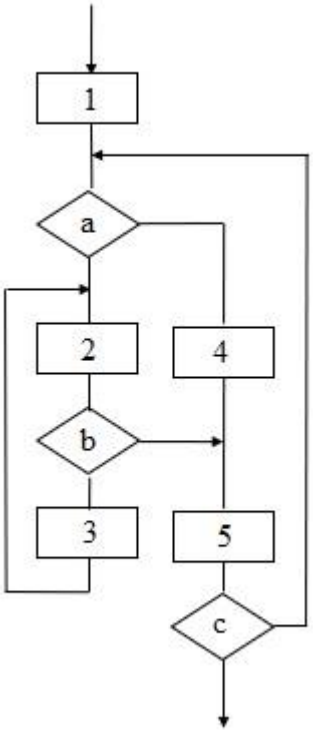
Вариант 42



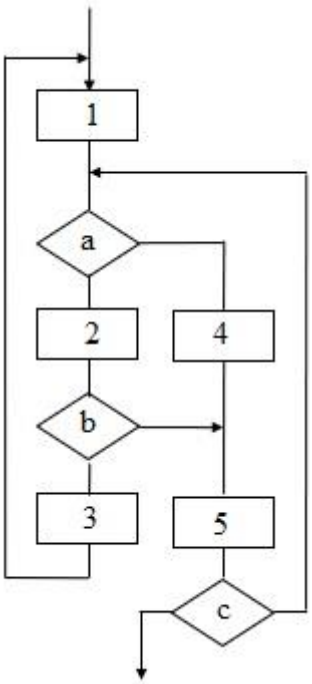
Вариант 43



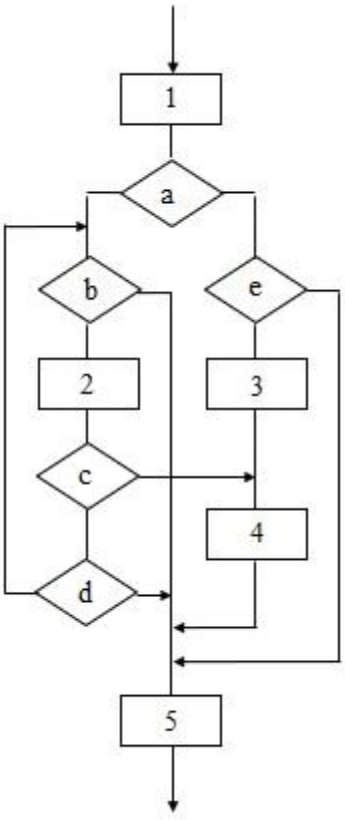
Вариант 44



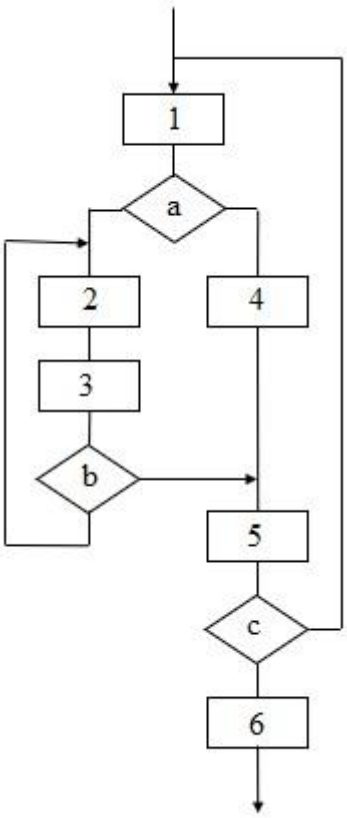
Вариант 45



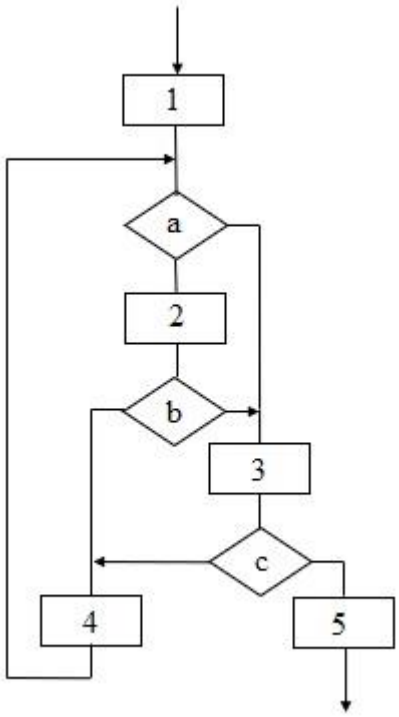
Вариант 46



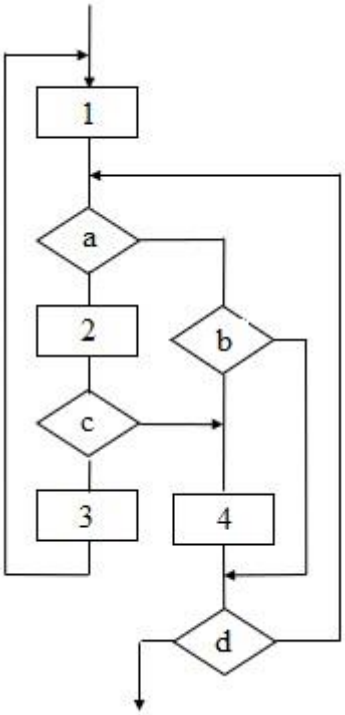
Вариант 47



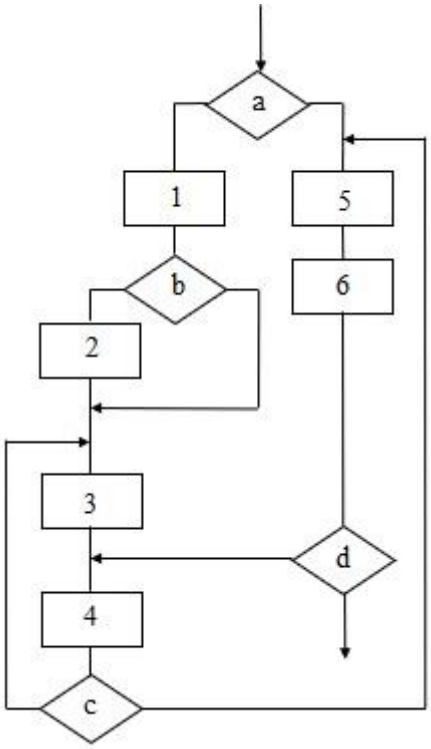
Вариант 48



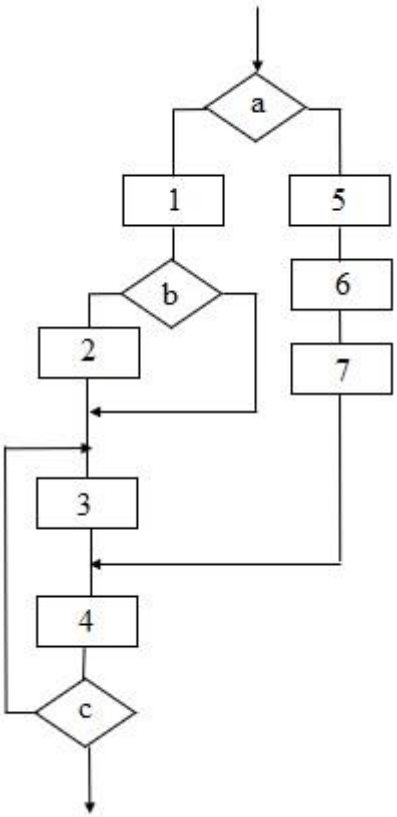
Вариант 49



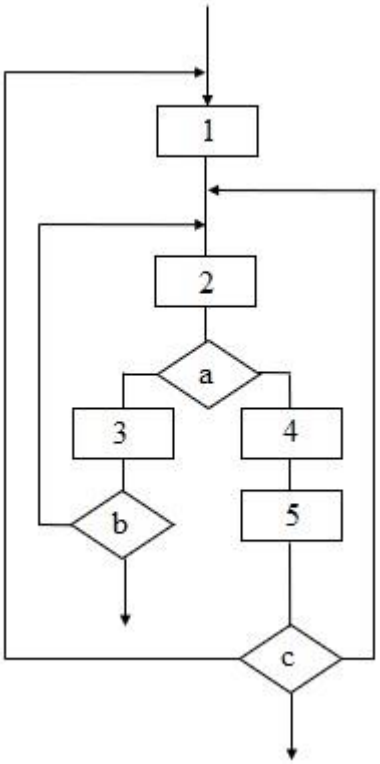
Вариант 50



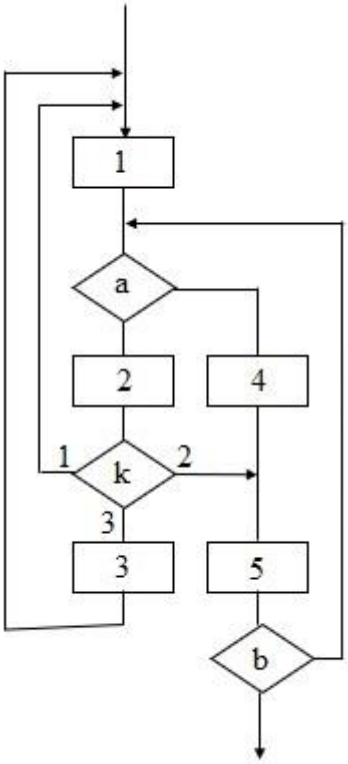
Вариант 51



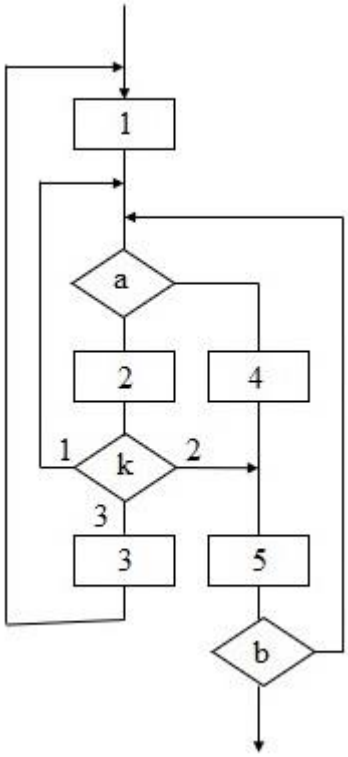
Вариант 52



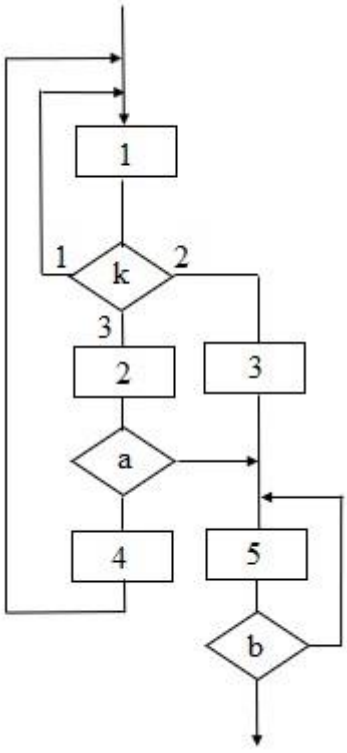
Вариант 53



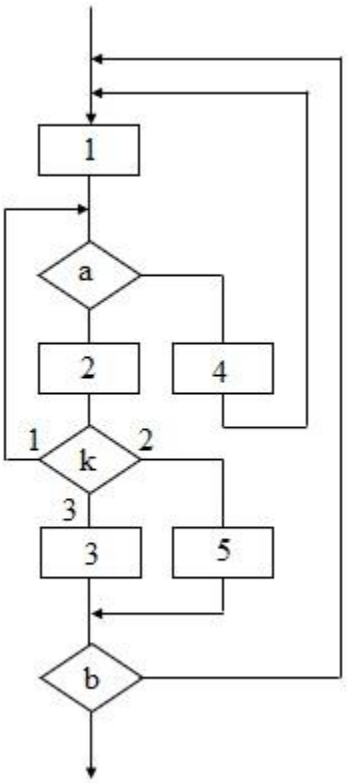
Вариант 54



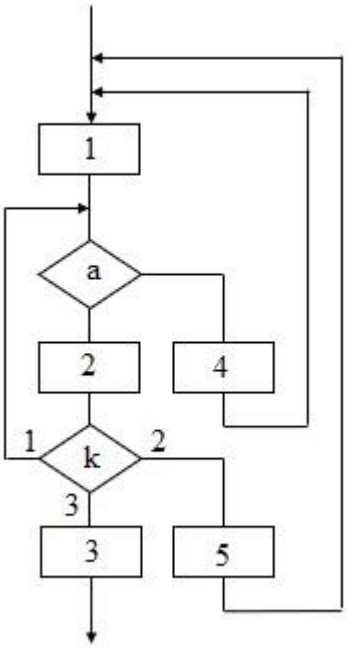
Вариант 55



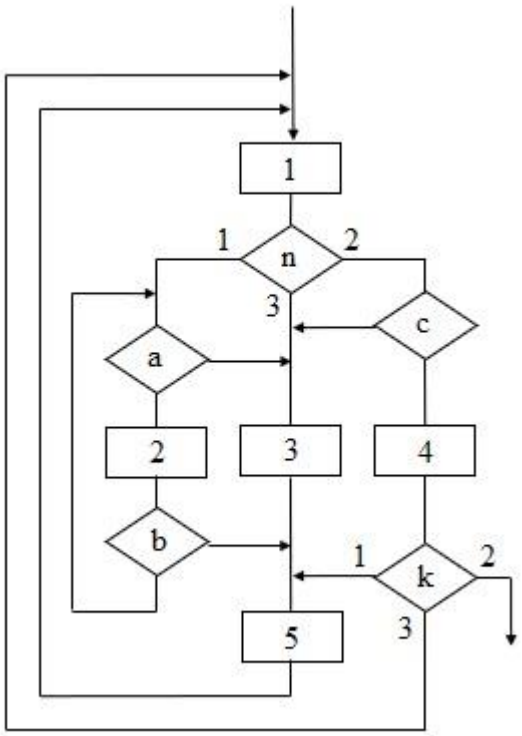
Вариант 56



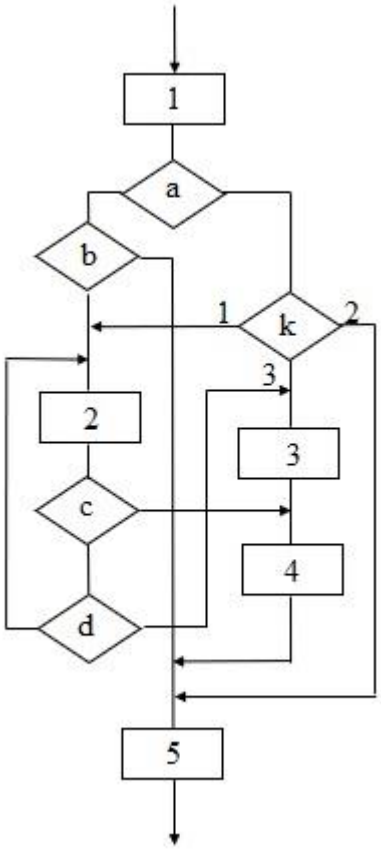
Вариант 57



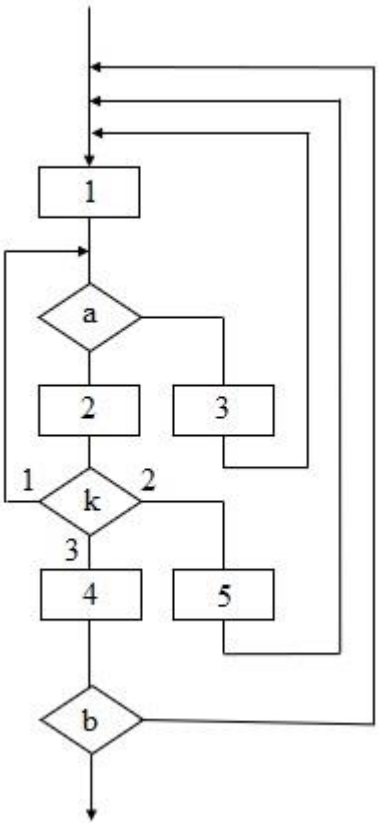
Вариант 58



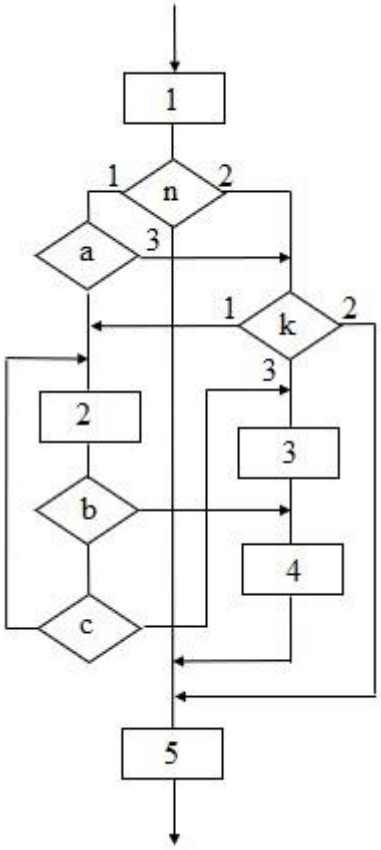
Вариант 59



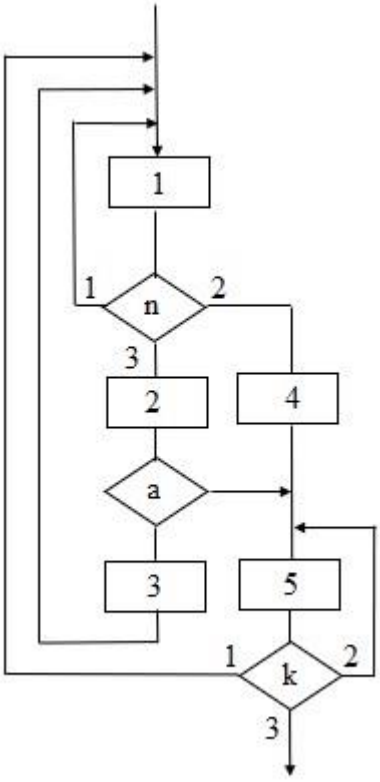
Вариант 60



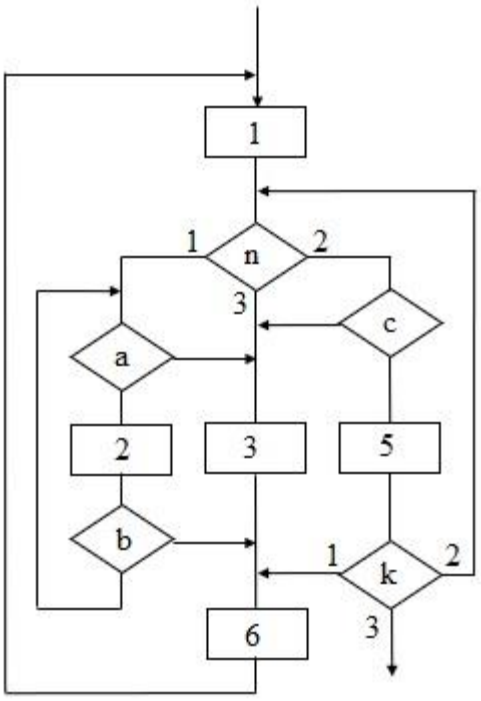
Вариант 61



Вариант 62



Вариант 63



Вариант 64

Содержание

Соничев Александр Викторович

Структурный рефакторинг диаграмм BPMN

Учебно-методическое пособие

Редактор Э.К. Долгатов

Подписано в печать . Формат 60×84 1/16
Бумага офсетная. Печать цифровая. Печ. л. 3,0.
Гарнитура «Times New Roman». Тираж 56 экз. Заказ

Издательство СПбЭТУ «ЛЭТИ»
197376, С.-Петербург, ул. Проф. Попова, 5