Жаринов В.Н.

АКТУАЛЬНЫЕ НАПРАВЛЕНИЯ ИНФОРМАТИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ

Вводный цикл (извлечение)

Версия 08.1

ВВЕДЕНИЕ В ДОКУМЕНТ

Общие положения

1. Файл содержит выполняемый автоматизированным способом (в форме электронного оригинала ЭО) [беловик|черновик] целевого документа или его части (неотъемлемой), выделенной для удобства работы.

Документ в целом, кроме основного содержания, может включать приложения. Содержание документа, приложения (его выделенной части) составляют текст и/или иллюстрации (графчасть).

Конкретное наполнение файла определяется по его имени (полный формат имен см. шаблон документа)[1].

2. Содержание документа, приложения подразделено на структурные элементы по иерархии; её высшие 4 уровня стандартны. Элементы обычно имеют многоуровневую нумерацию и заголовки-абзацы, входящие в оглавление; возможны также элементы без нумерации, в т.ч. не входящие в оглавление, в т.ч. с заголовками в тексте.

В тексте применяются типовые приемы оформления, описанные в п/р 1.1 документа|шаблона.

3. В файл части из документа, приложения выделяется элемент структуры стандартного уровня иерархии (или ряд соседних элементов одного уровня) целиком (с заголовками).

Для многофайлового ЭО в имени каждого файла указаны индексы входящих элементов (формат: разделы <ЧN>, подразделы <пPNN>, пункты <ППNNN>, подпункты <пПNNNN>); файл первой части является *головным*.

При наличии приложений их форму (способ выполнения) указывают в отметках о наличии в составе единственного (или головного) файла основного документа (виды способов и формат отметок см. шаблон).

Приложения в ЭО могут выполняться как отдельные файлы *ПрилN* (что указывается в их отметках о наличии).

При наличии иллюстраций в документе, приложении (части) они также м.б. выполнены разными способами. Подрисуночные подписи включаются в оглавление для удобства поиска рисунков в документе.

Иллюстрации в ЭО могут содержаться в отдельном файле графчасти *Рисунки*; тогда текст содержится в файле *Текст*, и в нём дублируется подпись к каждой иллюстрации по месту её упоминания для отсылки к графчасти.

4. Оригинал документа (части) выполнен как настоящий файл (имя см. поле внизу) и другие необходимые (детальный состав многофайлового документа см. п. 1.1.4 в <настоящем файле|головном файле *Ч.1 Введ.*>).

Текст подготовлен в среде OpenOffice.org 2.4.0 Writer или иной программы, совместимой по файлам; иллюстрации выполнены в той же программе и/или иными средствами, включая захват машобразов для ЭО.

Подлинник выполняется как твердая копия с заменой и/или добавлением листов к твердой копии предыдущих версий, либо как электронный образ файлов оригинала по листам, с которого делаются твердые дубликаты.

Информация, содержащаяся в документе, получена из открытых источников, рассматриваемых автором как надежные. Возможное наличие секретных, конфиденциальных, а равно иных сведений ограниченного доступа следует рассматривать как результат предположения на массивах открытых сведений. Имея в виду возможные человеческие и технические ошибки, автор не может гарантировать абсолютную точность и полноту приводимых сведений, и не несет ответственности за возможные последствия, связанные с их использованием.

---

[1] Переменные части текста даются как поля в '< >', заменяемые на описание; общая часть (корень) поля пишется как есть, а изменяемые части как '*'. Файлы ЭО с однокоренным именем относятся к одному элементу структуры.

[2] Федеральный закон № 230-ФЗ от 18 декабря 2006 г.

[3] Если цитата состоит полностью из сведений, цитирующих другой источник – дать ссылку на первоисточник.

Назначение, сведения о версиях, языковые соглашения

1. Документ содержит выделенные для совместной работы сводные части целевого документа и/или приложений к нему, указанные надзаголовками высшего уровня (с литерными индексами; см. также указания в п/п 1.1.1.3).

2. Сведения о назначении документа, его версиях, использованных обозначениях и сокращениях, а также общей структуре и источниках информации содержатся в головном файле <Ч.1 Введ.>.

3.В тексте документа употребляются следующие типовые обозначения и сокращения:

англ.	английский;
букв.	буквально;
в т.ч.	в том числе;
и т.д.	и так далее;
и т.п.	и тому подобное;
к.-л.	какой-либо;
напр.	например;
см.	смотри;
т.е.	то есть;
т. зр.	точка зрения;
т.о.	таким образом;
разд.	раздел (документа);
п/р	подраздел (документа);
п.	пункт (документа);
п/п	подпункт (документа);

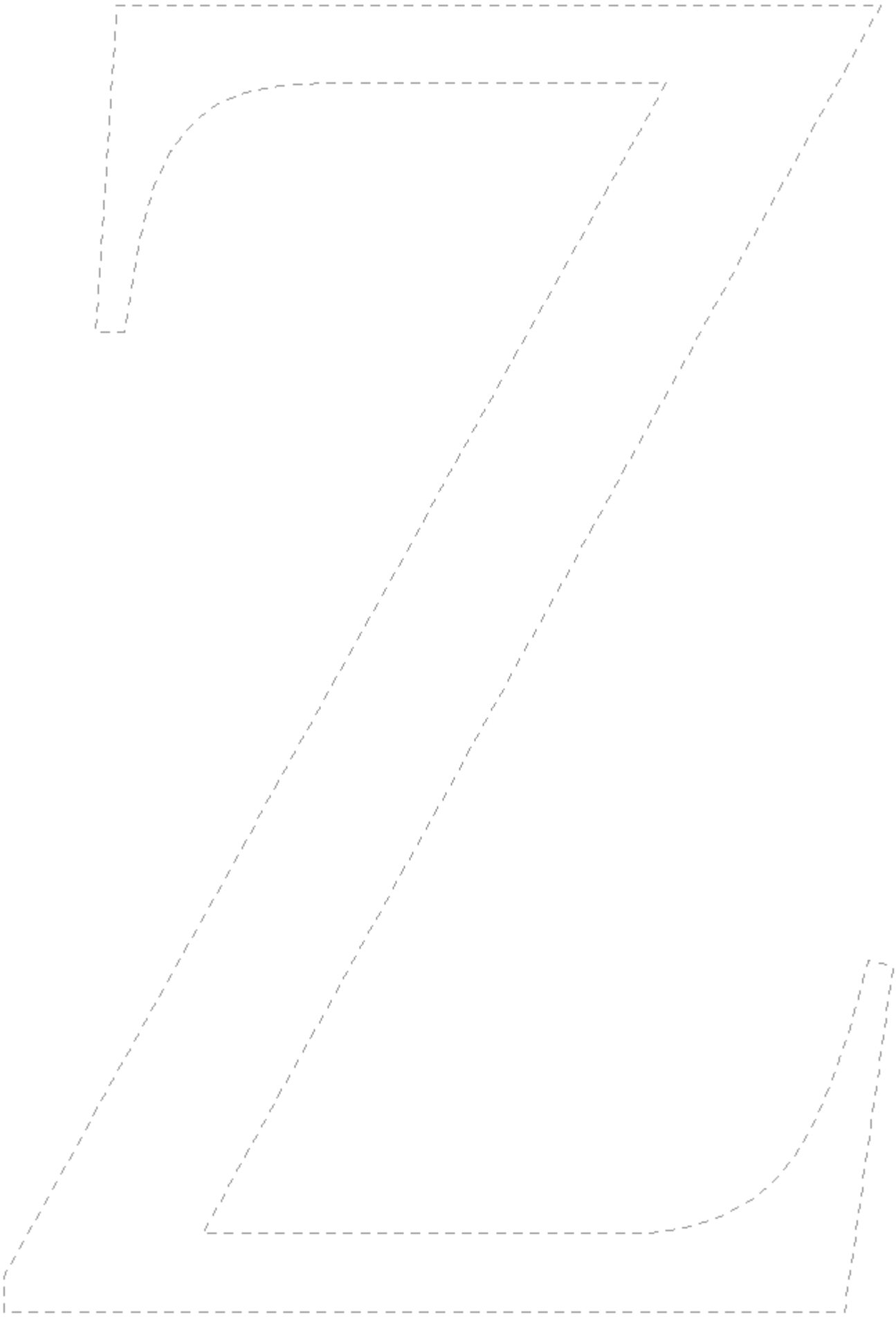
Об основных терминах данного документа см. п. 1.3.1.

4. Текущая версия черновика используется как источник беловика документа (в виде файла Writer с именем, включающим определённый номер редакции, или части такого файла при верстке) в составе готовых материалов по курсу; из беловика удаляются незавершённые элементы (разделы и пр.), ограничения для черновика и настоящий пункт, после чего оглавление пересобирается.

При необходимости работа над целевым документом/приложением (выделенной частью) отражается в служебном документе – плане (ЭО файл с приставкой <План>).

Оглавление

---[резервная страница оглавления]---



1. ВВЕДЕНИЕ

Внимание: следует хорошо изучить этот раздел, чтобы ориентироваться в документе.

1.1. Общие указания

1.1.1. В **тексте** документа применяются следующие приемы оформления.

1.1.1.1. Чтобы у***порядочить работу с материалом***, часть абзацев имеет особые стили:

- элементы перечисления оформляются как пункты маркированного списка;

Без красной строки оформляются абзацы, отбиваемые в объёмном тексте мысли для удобства чтения (за первым), а также вводные положения к крупному элементу (под заголовком).

Формулы обычным текстом даются центрованно отдельных строках

Такими абзацами оформлены фрагменты, на которые Вам следует обратить особое внимание (узловые моменты текущего пункта или наиболее важные выводы их него).

Внимание: [особое указание, требование, необходимое условие для применения пункта]

Эта информация дается сразу же после того места основного текста, к которому она относится.

– так выделяется пункт перечня, требующий особого внимания;

Так выделяется формула обычным текстом, требующая особого внимания

Пример. Такими абзацами выделяются примеры, иллюстрирующие текущую мысль основного текста. Так же выделяются практические рекомендации, советы, указания.

– так в тексте примера выделяется пункт перечня;

Так в тексте примера оформляется абзац продолжения текущей мысли.

Абзац с отступом и уменьшенным шрифтом выделяет в тексте документа составляющие развития, которые дополняют (уточняют, конкретизируют) содержание основного текста.

– так в тексте развития выделяется пункт перечня;

Так в тексте развития оформляется абзац продолжения текущей мысли.

Так в тексте развития выделяется формула

Информация к размышлению. [подзаголовок статьи]. Таким образом в тексте выделена познавательная составляющая, которая не является обязательной для изучения, но может расширить и углубить понимание предмета.

- так в тексте отступления выделяется пункт перечня;

Так в тексте отступления оформляется абзац продолжения текущей мысли.

1.1.1.2. С той же целью фрагменты в тексте могут оформляться в следующих стилях:

Жирным шрифтом выделены названия отдельных пунктов, уровни классификации или комментарии в тексте к элементам схем, диаграмм.

Курсивом выделяются понятия, определяемые в тексте документа, а также предложения, содержащие важную информацию (выводы, указания и пр.). В списке литературы курсивом выделены позиции, которые имеются в [учебной|служебной] библиотеке.

Жирным курсивом выделены подуровни классификации либо понятия, о которых идет речь в окружающем тексте, или которые уже должны быть Вам известны.

Подчеркиванием обозначаются ссылки на место в данном документе или за его пределами, напр., на другие документы (кроме гиперссылок на ресурсы интернет, которые оформляются стандартно для эледоков). Если подчеркнута ссылка на другие дисциплины, сферы деятельности, то Вы можете обратиться за информацией к соответствующим специалистам, преподавателям, в интернет.

Разрядкой выделены места, на которые следует обратить особое внимание.

Таким начертанием (гарнитурой) шрифта и курсивом выделены наименования объектов (сущностей), описываемых в документе.

Такой гарнитурой шрифта (с уплотнением) выделены тексты процессов (алгоритмов, программ).

Так выделяется внутритекстовый заголовок части пункта (подпункта). Далее начинается собственно текст этой части. Такой заголовок не входит в оглавление документа.

1.1.2. В **графической части** документа используются стандартные стили текста и условные обозначения, приведённые далее (см. п. 1.3.1).

Иллюстрации упорядочены по тексту и снабжены подписями вида:

Так оформляется подрисуночная подпись

В файле выделенного текста эти подписи указывают наличие и положение иллюстраций.

> **Внимание**: отдельные подписи могут размещаться не под, а над рисунком. В любом случае подпись относится к тому рисунку, к краю которого она расположена вплотную.

1.1.3. В тексте **черновика** (**плана**) применяются также иные приемы оформления:

Такими абзацами (уровня 6 в иерархии элементов текста) оформляется краткое содержание текста текущего элемента структуры в черновике (плане) .

Замечание. Так оформляются замечания автора по содержанию текста.

1.1.4. Черновой документ (план) м. б. сводным из частей для разных целевых документов, совместно разрабатываемых по единому замыслу, а в беловом могут присутствовать части, которые исходя из назначения документа должны выделяться в самостоятельные целевые документы. Для таких сводных частей дополнительно вводится уровень надзаголовков вида:

{X}[N]. <Текст надзаголовка>

Здесь: X - литер целевого документа (в плане/черновике для группы документов, приложений);
N - номер самостоятельной сводной части, выделяемой из документа, приложения.

> В сводную часть черновика могут входить сводные части беловика. Надзаголовки последних находятся в границах первой (указываются как неформальные комментарии в абзацах уровня 5).

---конец служебной части (введения в документ) для документа/шаблона---

1.2. Общие положения

1.2.1. В документе излагается суть произвольной задачи и процесс решения этой задачи (технология формализации знаний о решении).

Документ предназначен для включения в курс лекций по информатике.

Задача поставлена в общем виде (независимо от содержания) и описана в стандартной для курса форме (по формуляру), а техпроцесс формирования решения — как базовая ТФЗ.

1.2.2. Положение каждой сводной части в курсе указано в начале (после надзаголовка).

1.2.3. Документ подготовлен с использованием источников информации, указанных в Разд. В. Необходимая информация содержится также в приложениях к документу:

1.2.4. Основное содержание документа в оригинале выполнено как настоящий файл.

Приложения к документу выполнены как отдельные файлы (см. отметки о наличии).

1.3. Необходимые определения

1.3.1. Основные текстовые и графические термины, обозначения и сокращенияданного документа введены в п/р 1.3, а другие необходимые — в тексте п/р 1.4 курса, а также определяются в его тексте и визуально на рисунках по ходу изложения.

1.3.2. Здесь расшифрованы сокращения, часто употребляемые в тексте документа.

> Сокращения, употребляемые лишь в отдельных местах текста, расшифровываются там же.
> Сокращения, значащие одно и то же (напр., на разных языках), отсылают друг к другу.

БНФ	Бэкуса-Наура формы <определения синтаксиса текстов>
ГСА	граф-схема алгоритма
ДРАКОН	Дружелюбный Русский Алгоязык, Который Обеспечивает Наглядность

A. Вводный (опорный) материал по алгоритмике

---включается в основной текст лекций согласно заголовкам, предшествующим началу сводной части. Индекс [N] устанавливается по порядку пункта в пределах подраздела--

МЕТОДИКИ СОСТАВЛЕНИЯ АЛГОРИТМОВ НА ТЕХНОЯЗЫКЕ

Вводные замечания

Укрупнённо процесс алгоритмизации с применением техноязыка ДРАКОН описан как технология формализации знаний в Разд.4. Очевидно, что детализация ТФЗ-ДРАКОН приводит к появлению методик эскизной и детальной визуализации; в перспективе должен формироваться комплекс таких методик.

Комплекс в целом м.б. рассчитан на сложившиеся пути автоматизации, формализующее знания путем создания спецификаций задач пользователя и разработки программы решения каждой задачи. Программа может выполняться изолированно или взаимодействовать с другими программами (как встроенная процедура пользователя или часть комплекса программ, сопряжённая с другими только по входным/выходным данным).

В типичном случае программа составляется на основных алгоритмических языках. Как правило, это Си, Паскаль, Бейсик или макроязыки пакетов прикладной автоматизации (обозначаемые общим именем X, как при описании техноязыка). Техническое обеспечение составляют офисные системы для подготовки спецификаций задач (в первую очередь текстов программ автоматизации решения) и системы программирования для получения машинного кода программ. Поэтому предполагается, что спецификации задач готовятся в текстовом процессоре с возможностью рисования.

Другой типичный случай – автоматизация путем разработки программы для СУБД (или включения новых функций в уже существующую) – отличается главным образом тем, что в качестве формального декларативного языка вместо выбранного стандарта используется стандарт графического моделирования данных вида «сущность-связь», например по методологии IDEF1X. При этом текстовый процессор обычно заменяется на редактор моделей данных, а в качестве языка X выступает язык описания запросов к БД (например, SQL).

В любом случае методики входят как часть в технологический процесс гибридного программирования, результатом которого является программный код на языке X и визуалы-инструкции пользователю.

В описании методики структура процессов задается многоуровневой нумерацией элементов текста, а ветвления и циклы указываются ссылками на номера; элементарные шаги следования выделяются как элементы списка со специальными пометками (на техноязыке соответствуют операторам внутри шампур-блока структурного типа). Как и в техноязыке, при отсутствии переходов принят естественный порядок следования от пункта к пункту; поэтому при описании развилок даются только указания на переходы, нарушающие этот порядок.

Вообще говоря, для методики необязательна алгоритмическая строгость процесса, поэтому в основном описания будут процедурными.

Исполнителем методики считается человек с обычным уровнем подготовки; обязательно знание техноязыка и основ моделирования данных, а также методов решения задач в своей предметной области.

Визуал простой задачи при некотором навыке можно строить мысленно; однако предполагается, что визуализация протекает с документальным оформлением и редактированием полученного документа (натурального или машинного).

Методика визуализации алгоритмов переработки данных

Методика применяется для построения визуального алгоритма на основе неалгоритмического описания задачи. Требования к процессу визуализации здесь ослаблены путем выбора

менее строгих значений управляющих факторов; в первую очередь это касается целостности и свободы оформления, что важно для творческой работы.

Методика основана на положениях технологии формализации знаний с помощью техноязыка ДРАКОН, а также на методической разработке процесса алгоритмизации В. Ляховичем[4].

Эта разработка позволяет получить алгоритм преобразования исходных данных в результатные с формальным текстом икон, который пригоден для гибридного программирования (кодирования на языке ДРАКОН-Х для исполнителя-машины). Она также обладает рядом достоинств, как-то:

- отработана автором на практике в течение значительного времени (на момент опубликования – порядка 15 лет);
- проста в применении и эффективна как средство автоформализации и профессиональных, и информатических знаний за счет обобщённой структуры, в то же время открытой для уточнения исполнителем по мере накопления собственного опыта;
- излагается с минимальным уровнем формализации и математизации, что соответствует объективно возможному на сегодня массовому уровню математической и информатической культуры предполагаемых исполнителей методики (студентов, слушателей курсов переподготовки);

Тем не менее методика Ляховича потребовала корректировки по следующим причинам:

- разработка выполнялась для основного на тот момент «школьного» языка блок-схем и требует перевода на техноязык (впрочем, вполне тривиального);
- для исполнителей с учетом возможного уровня их информатической культуры нужно явно описать некоторые условия и допущения, принятые Ляховичем.

Результат корректировки оформлен как частная методика (см. в Приложении 8).

Также методика Ляховича д.б. включена в более общую методику визуализации как часть (необходимая) по следующим причинам:

- за её рамками остается работа оператора, в частности организация его диалога с машиной, и определение данных, в т.ч. откуда берутся исходные величины и куда деваются результатные;
- не определяется использование результатов алгоритмизуемого процесса.

В качестве наиболее общей методики, внешней по отношению к данной (включающей её как часть, «охватывающей»), выступает ТФЗ-ДРАКОН в определении, данном здесь. На основе этой технологии м.б. разработаны конкретные методики визуализации.

Поскольку структура визуала обработки данных задачи по частной методике задается сначала и в дальнейшем не изменяется, то допускается, что на завершающих шагах исполнитель может обнаружить неоптимальность этой структуры (эмпирически или на основе к.-л. формальных критериев). В этом случае частная методика может применяться сначала, но с выбором иной структуры задачи, т.е. происходит оптимизация по методу «что будет, если…».

Заключение

В целом методика охватывает этапы 1 и 3 ТФЗ-ДРАКОН.

Достигаемый промежуточный результат – исходный чертёж (дракон-программа) решения задачи на гибридном языке ДРАКОН-Х.

Исполнитель визуала – любой субъект, обладающий следующими характеристиками:

- Множество типов данных (будем называть его *багажом*) образуют скалярные и/или векторные величины, как определено в частной методике; фактически речь идет о супертипах простых и составных величин.
- Множество предписаний исполнителя (т.н. *репертуар*[5]) составляют любые действия над этими величинами, представленные в форме арифметических и логических выражений.
- Множество механизмов (т.н. *реквизит*) составляют средства выполнения арифметико-логических операций и операторов управления дракон-схем.

[4] Ляхович В. Методика составления алгоритмов. - Информатика и образование, №1/1990. - С.39-47.

[5] По терминологии автора техноязыка.

- Языком общения исполнителя является гибридный техноязык ДРАКОН-X, где командный подъязык X выбирает исполнитель методики (далее – сочинитель).

Предполагается, что исполнителем м.б. как человек, знакомый с техноязыком ДРАКОН, так и машина, для которой составлена программа на языке X.

Отметим вопросы, которые должны решаться на последующих этапах ТФЗ с учетом свойств получаемого результата визуализации.

Любой реальный язык программирования X обладает множеством типов данных как скалярного, так и векторного супертипов и правилами их совместного употребления в выражениях, а также преобразования (т.н. *переопределения* типов); вопросы рациональной типизации решаются при кодировании дракон-программы, т.е. на этапе 4 ТФЗ.

При отсутствии автоматического конвертора исходных чертежей с языка ДРАКОН-X код (исходный текст) на языке X можно получить только вручную.

Примеры визуализации алгоритмов обработки данных

Далее описаны составленные Ляховичем примеры применения методики алгоритмизации к конкретным задачам.

Мы дополнительно будем указывать возможности, следующие из каждого примера. Задачи будем формулировать, как определено в лекциях (в виде тройки «дано, надо, решение»).

Вывод формул и доказательство их справедливости

Задача 1

Дано: последовательность элементов $a_1, a_2, a_3, ...,a_n$,

где n чётное.

Надо: определить сумму элементов последовательности с нечётными номерами S и визуализировать соответствующий алгоритм.

Решение (только для вывода рабочих формул).

Понятно, что объект исходных данных в задаче один - массив элементов последовательности a_i. К промежуточным величинам относится i, n; к результатным – S.

Описание процесса решения задачи по этапам:

1) $S_1=a_1+a_3$;

2) $S_2=S_1+a_5$;

3) $S_3=S_2+a_7$.

Из сопоставления формул трех этапов несложно подметить, что будет справедливо инвариантное выражение

$$i)\ S_i=S_{i-1}+a_{2i+1} \qquad (3)$$

Это выражение и будет основной рабочей формулой.

Последний этап описывается как:

$$S_r=S_{r-1}+a_{n-1}$$

Приводя операцию первого этапа к инвариантному виду, имеем:

$$S_1=S_0+a_3$$

Отсюда следует, что $S_0=a_1$, т.е. найдено начальное значение S.

Кроме S в формуле (3) используется i - номер этапа. Начальное значение i=1 определяется из (3), закон изменения

$$i:=i+1 \qquad (4)$$

Выражение (4) - это формула изменения аргумента. Других переменных в задаче нет.

Из того, что индекс величины a в (3) не может превышать n-1, вытекает условие ПЦ:

$$2i+1\le n-1, \text{ или } i\le(n-2)/2.$$

Докажем справедливость выражения (3).

При i=1 выражение верно, что очевидно.

Пусть (3) верно при некотором i=k; покажем, что оно верно и для следующего числа k+1:

$$S_k=S_{k-1}+a_{2k-1}=a_1+a_3+...+a_{2k+1},$$

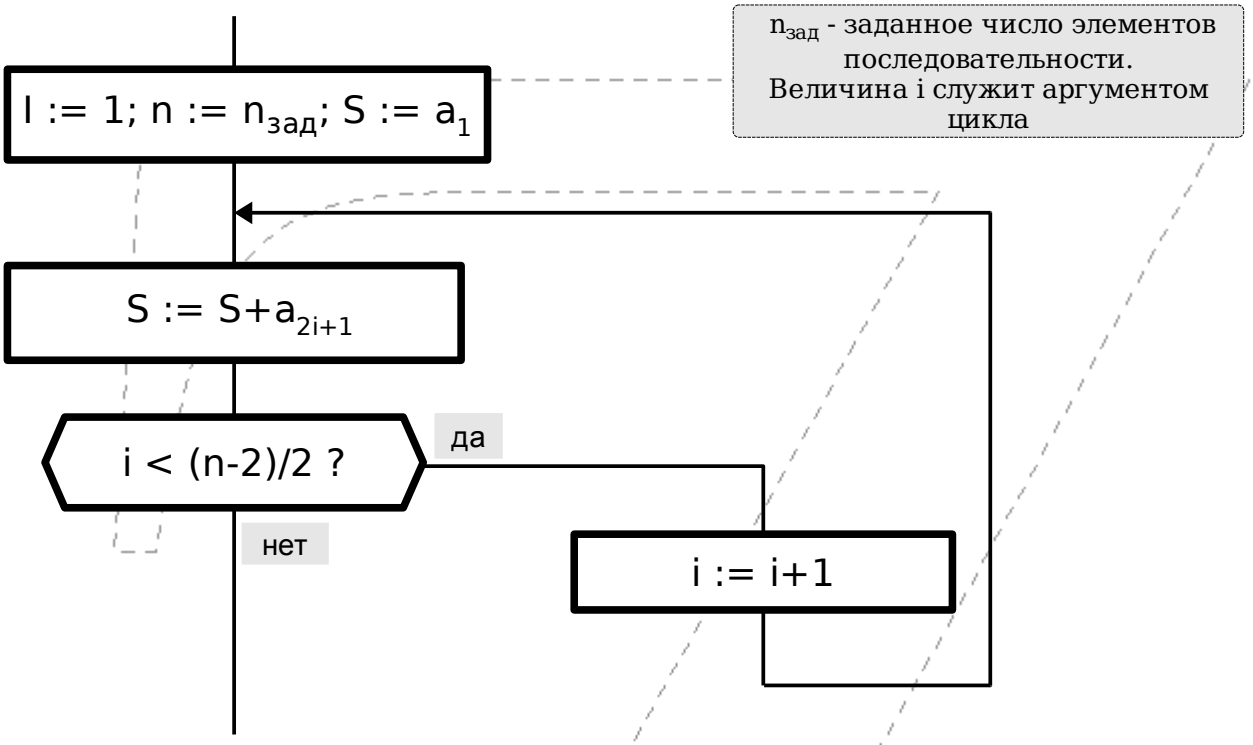
$$S_{k+1}=a_1+a_3+...+a_{2k+1}+a_{2k+2+1}=S_k+a_{2k+2+1},$$

т.е.

$$S_{k+1}=S_k+a_{2(k+1)+1}$$

Мы получили (3), где вместо i стоит k+1; следовательно, на основании аксиомы математической индукции заключаем, что (3) верно для любого натурального числа i.

Определены все объекты циклического алгоритма. Заполнив ими блоки типовой схемы, получаем решение поставленной задачи. Дракон-схема визуала приведена далее (см. графчасть)



Визуализация содержания алгоритма решения задачи 1

Отметим, что если n нечетное, то задача решается с использованием полученного для задачи 1 результата в два этапа:

1) дополнение массива a до четного числа элементов (нулевым элементом);

2) определение S по построенному выше алгоритму.

Т.о., такая задача уже относится к произвольному виду.

Составление алгоритмов с одним циклом

Здесь и далее мы будем описывать алгоритмизацию по шагам соответствующей методики. Доказательства справедливости получаемых формул не приводятся, поскольку этот процесс весьма схожий для всех задач.

Величины и выражения будем записывать, как принято в программировании, а именно:

- массив и элемент массива обозначаем одинаковым именем, заключая индекс в квадратные скобки, а для ряда измерений перечисляя индексы через запятую (в отличие от индексирования переменных по этапам решения задачи, которое будем оформлять по-прежнему);
- операцию умножения обозначаем явно знаком '*'.

Для некоторых величин верхняя граница диапазона значений м.б. неизвестна заранее; в этом случае обозначаем её знаком '?'. Эта ситуация не влияет на детерминированность алгоритма; при исполнении конечное число его шагов обеспечивается точным заданием условия ПЦ (ОПЦ) от определённых текущих значений величин.

Рабочий блок результирующего визуала в общем случае имеет сложную внутреннюю структуру (подробную схему), напр. разветвляющуюся; для наглядности на результирующих схемах будем выделять его пунктиром.

Имена величин (кроме служебных) можно сделать информативными, давая их по общеязыковым правилам; здесь мы сохраним именование Ляховича.

Задача 2

Дано: функция (многочлен) $Z=3X^5-X^4+6X^3-2X^2-7X+3$

Надо: вычислить Z при одном значении X.

Решение.

1. Принимаем формулировку условия задачи.

2. Описываем объекты данных.

2.1. К исходным данным относятся:

- X (постоянная для данной задачи величина)
- множество коэффициентов многочлена; закон их изменения отсутствует, следовательно составим из них массив A=<3, -1, 6, -2, -7, 3>=A[0], ..., A[5]

2.2. К результатам относятся:

Z – значение функции.

3. Метод решения очевиден и состоит в вычислении выражения для Z. Этот процесс можно разбить на этапы, на каждом из которых определяется Z с учетом члена очередной степени.

5. Выводим рабочие формулы.

5.1. Порядок решения задачи.

этап 1: b_1:=A[4]*X+A[5].

этап 2: r_2:=X*X; b_2:=A[3]*r_2+b_1.

этап 3: r_3:=r_2*X; b_3:=A[2]*r_3+b_2.

Здесь вводятся промежуточные величины:

r – произведение X на предыдущее r, т.е. очередная степень X при текущем члене многочлена;

b – сумма текущего члена с предыдущими (нарастающий итог, на последнем этапе становящийся результатом).

5.2. Обобщение решения.

этап i: r_i:=r_{i-1}*X; b_i:=A[5-i]*r_i+b_{i-1}

5.3. Формулировка последнего этапа.

этап k: r_k:=r_{k-1}*X; b_k=Z:=A[0]*r_k+b_{k-1}

5.4. Для единообразия 1-ю операцию этапа 2 запишем так:

r_2:=r_1*X, откуда r_1=X.

Этап 1 дополним операцией над r_i и приведем к обобщённому виду:

r_1:=r_0*X; b_1:=A[4]*r_1+b_0,

откуда следуют начальные значения: r_0=1 (с учетом предыдущего); b_0=A[5].

6. Окончательно формируем содержание икон для циклического алгоритма.

6.1. Выделяем переменные и выражения (рекуррентные) для них:

r:=r*X |r=1...?

b:=A[5-i]+b |b=1...?

i:=i+1 |i=1...5

Индексы при переменных r, b опускаем.

Отказ от индексации переменных вытекает из логики выполнения оператора присваивания, когда в нем участвуют лишь текущее и предшествующее значения величины по разные стороны знака присваивания: до выполнения существует значение в правой части, а после – в левой.

Вертикальной чертой здесь и далее отделяем указание диапазона изменения переменной.

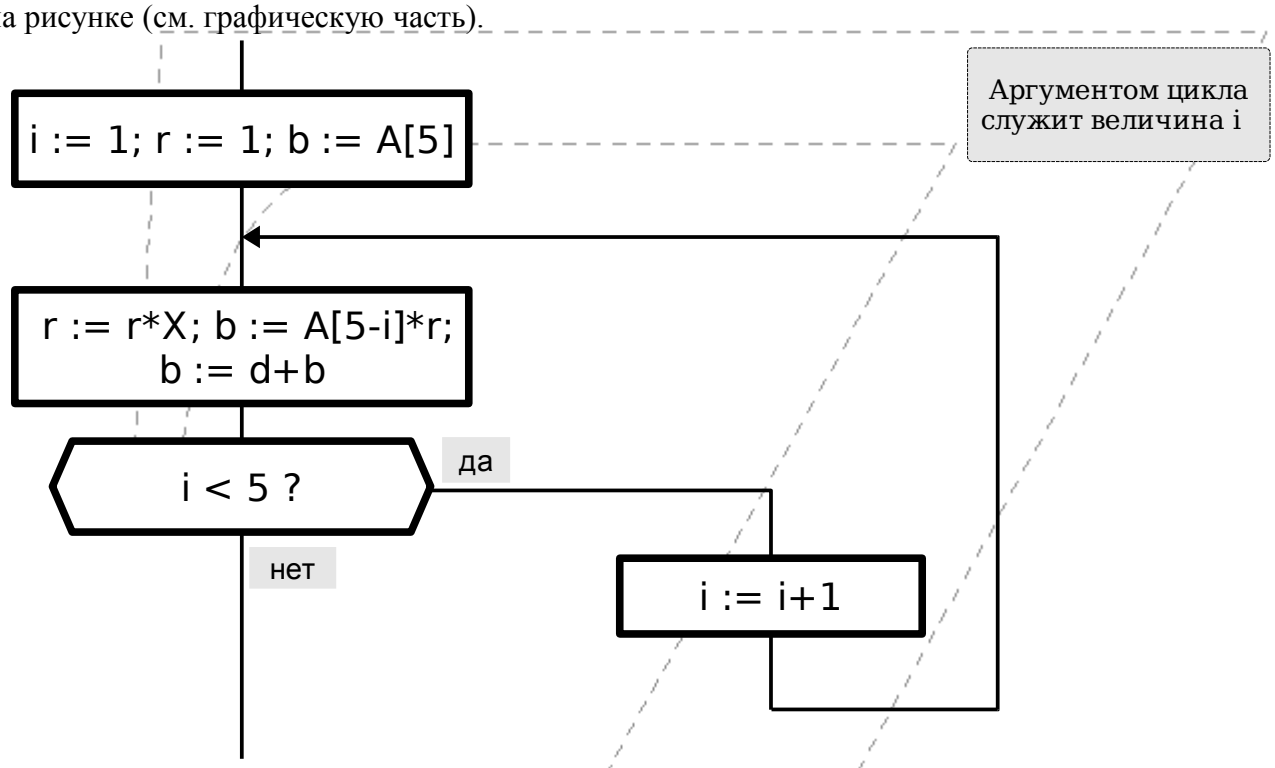
При вычислении b вводим промежуточную величину d и разбиваем выражение на два:

d:=A[5-i]*r ; b:=d+b.

Прием вычисления через промежуточные величины в данном случае упрощает выражения (что необязательно, если только язык X не ограничивает число операций в одном выражении); далее он будет применяться и с другими целями.

6.2. Из сравнения индексов при A на i-том и k-том этапах вытекает условие ПЦ: i<5.

7. Изображаем типовую схему шампур-блока и заполняем её иконы. Результат показан на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения задачи 2

Отметим, что на основе полученного решения можно решить и более общую задачу вычисления функции при ряде значений X. Для этого X определяется как массив и составляется укрупнённая схема с циклом по элементам X; рассмотренное решение в этой схеме фигурирует как содержание рабочего блока цикла. Т.о. имеем произвольный алгоритм со вложенными циклами.

Задача 3

Дано: матрица A(nXn).

Надо: найти наибольший элемент побочной диагонали.

Решение.

1. Принимаем формулировку условия задачи.

2. Описываем объекты данных.

2.1. К исходным данным относятся:

n – размерность матрицы (постоянная для данной задачи величина);

матрица A(nXn).

2.2. К результатам относятся:

b – значение наибольшего элемента побочной диагонали.

3. Метод решения: сравниваем первые два элемента диагонали и находим наибольший. Результат сравниваем с третьим элементом диагонали и находим наибольший и т.д.

5. Выводим рабочие формулы.

5.1. Порядок решения задачи.

этап 1: проверить условие A[1,n]>A[2,(n-1)]; если оно выполняется, то наибольшее на данном этапе значение b_1:=A[1,n], иначе b_1:=A[2,(n-1)].

этап 2: проверить b_1>A[3,(n-2)]; если выполняется, то b_2:=b_1, иначе b_2:=A[3,(n-2)].

этап 3: проверить b_2>A[4,(n-3)]; если выполняется, то b_3:=b_2, иначе b_3:=A[4,(n-3)].

5.2. Обобщение решения.

этап i: b_{i-1}>A[(i+1),(n-i)]? : да – b_i:=b_{i-1}; нет – b_i:=A[(i+1),(n-i)].

Здесь мы одновременно перешли к формулировке текста условия в виде вопроса с «да-нетным» ответом, как принято в техноязыке; после тире перечисляются операторы вертикали развилки, соответствующей этому ответу. Этой записи будем и далее придерживаться в условных конструкциях.

5.3. Формулировка последнего этапа.

этап r: b_{r-1}>A[n,1]? : да – b_r:=br-1; нет – b_r:=A[n,1].

5.4. Для единообразия операции этапа 1 запишем так:

b_0>A[2,(n-1)]? : да – b_1:=b_0; нет – b_1:=A[2,(n-1)],

откуда b_0=A[1,n],

6. Окончательно формируем содержание икон для циклического алгоритма.

6.1. Выделяем переменные и выражения (рекуррентные) для них:

b>A[(i+1),(n-i)]?: да – b:=b; нет – b:=A[(i+1),(n-i)] |b=A[1,n]...?

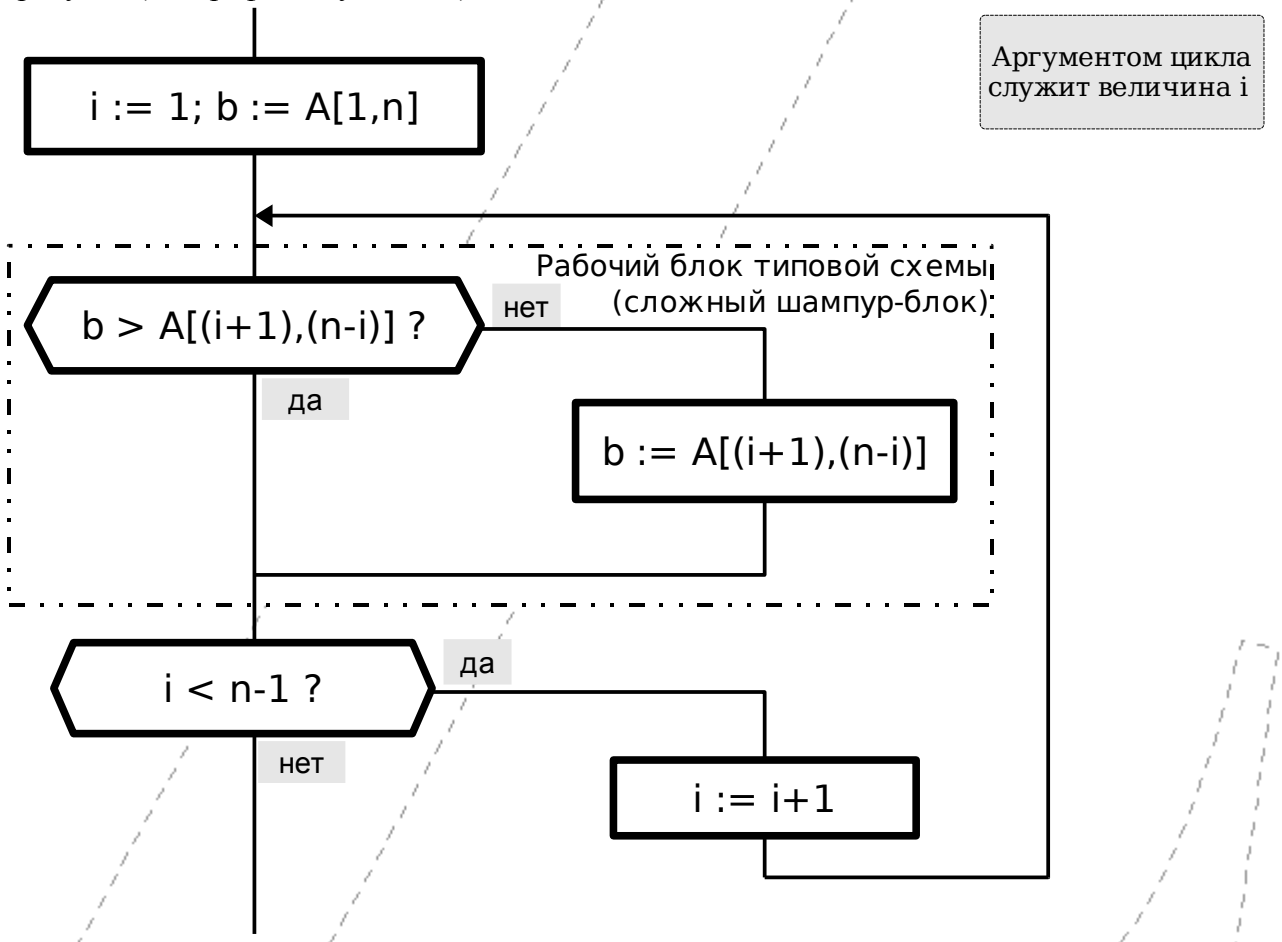
i:=i+1 |i=1...n-1

Индекс при b опускаем, т.к. для вычисления любого последующего b достаточно знать лишь его предшествующее значение, а оно содержится в выражении для присваивания; это выражение имеет два варианта, из которых на каждом этапе выбирается один по условию.

6.2. Из сравнения индексов при a на i-том и k-том этапах вытекает условие ПЦ:

i+1<n, или i<n-1.

7. Изображаем типовую схему шампур-блока и заполняем её иконы. Результат показан на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения задачи 3

Задача 4

Дано: функция $Y=a^X$ (X<0)

Надо: вычислить приближённое значение функции с помощью разложения её в ряд:

$Y=1+X*\ln a/1+X^2*\ln^2 a/2+...$

Допустимая погрешность - e.

Решение.

1. Для алгоритмизации формулировку условия задачи нужно дополнить общими сведениями из математики о решении подобных задач, а именно:

- за приближённое значение функции принимается сумма n членов ряда;
- погрешность b такого приближения принимается равной значению n-го члена ряда.

2. Описываем объекты данных.

2.1. К исходным данным относятся:

a, X, e (постоянные для данной задачи величины); b (переменная текущая погрешность).

2.2. К результатам относятся:

Y (одно значение).

3. Метод: последовательное вычисление каждого члена ряда, начиная с первого, и сложение его с предшествующей суммой.

Введём обозначение (промежуточную величину): c=ln(a);

Как и в предыдущей задаче, промежуточная величина упростит запись выражений; кроме того, тем самым мы косвенно «вводим в обращение» исходную величину a. Однако важнее, что здесь этим приёмом мы избегаем многократного логарифмирования в ходе исполнения алгоритма.

5. Выводим рабочие формулы.

5.1. Порядок решения задачи.

этап 1: $b_1:=c*X$; $Y_1:=b_1+1$.

этап 2: $r_2:=1*2$; $b_2:=b_1*c*X/r1$; $Y_2:=b_2+Y_1$.

этап 3: $r_3:=r_2*3$; $b_3:=b_2*c*X/r_3$; $Y3:=b_3+Y_2$.

Здесь промежуточная величина r - это произведение знаменателя текущего члена ряда на знаменатели предыдущих; такое определение позволяет построить рабочую формулу для b.

5.2. Обобщение решения.

этап i: $r_i:=r_{i-1}*i$; $b_i:=b_{i-1}*c*X/r_i$; $Y_i:=b_i+Y_{i-1}$.

5.3. Из п.1 известно условие ОПЦ будущего алгоритма: |b<e|, поэтому формулировка операций последнего этапа не требуется.

5.4. Для единообразия операции этапов 1 и 2 запишем так:

$r_2:=2*r_1$; $r_1:=1*r_0$; $b_1:=b_0*c*X/r_1$; $Y_1:=b_1+Y_0$.

Отсюда следует, что начальные значения r, b, Y равны 1.

6. Окончательно формируем содержание икон для циклического алгоритма.

6.1. Выделяем переменные и выражения (рекуррентные) для них:

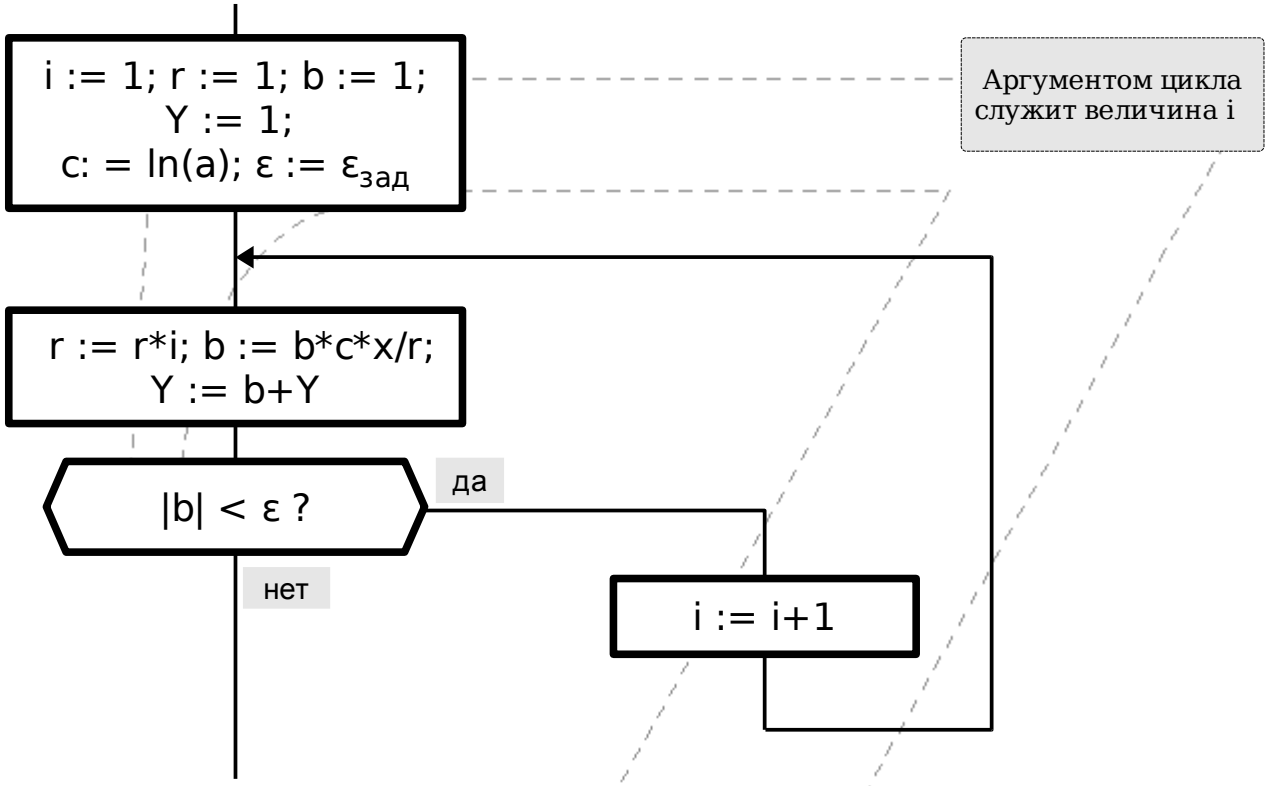
i:=i+1 |i=1...?

Y:=Y+b/r |Y=1...?

r:=r*i |r=1...?

b:=b*c*X/r |b=1...e

7. Изображаем типовую схему шампур-блока и заполняем её иконы. Результат показан на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения задачи 4

Задача 5

Дано: массив A(1:n).

Надо: выбрать из массива все положительные элементы.

Решение.

1. Принимаем формулировку с уточнением: результат оформим также в виде массива (исходя из правила выделения и типизации данных).

2. Описываем объекты данных.

2.1. К исходным данным относятся:

n – размерность массива; A[1…n].

2.2. К результатам относятся:

B[1…n].

Размер массива B максимален, т.к. число положительных элементов массива A заранее неизвестно.

3. Метод: сравниваем с нулем каждый элемент массива A по порядку, начиная с 1-го; если его значение больше нуля, то переносим его в массив B.

5. Выводим рабочие формулы.

5.1. Порядок решения задачи «в лоб» можно описать так.

этап 1: проверить A[1]>0; если выполняется, то B[1]:=A[1], иначе переход к следующему этапу.

этап 2: проверить A[2]>0; если выполняется, то B[2]:=A[2], иначе переход к следующему этапу.

Однако такое построение ведет к ошибке в решении: элементы массива B, индекс которых соответствует индексу неположительных элементов массива A, будут обойдены в ходе исполнения алгоритма; на практике они будут сохранять значения на момент выделения памяти под массив B[6].

Правильным будет считать, что индекс очередного элемента массива B не совпадает с номером текущего этапа; в общем случае на i-том этапе он может принимать значение от единицы до i.

Исходя из вышесказанного, индекс для B обозначаем как переменную величину f.

[6] Ошибка происходит потому, что «лобовой» алгоритм игнорирует неявное требование к результату: элементы массива B должны следовать непрерывно по порядку начиная с 1-го, как и в массиве A; тем самым результат отделяется от неиспользуемой части массива B; отсюда видно, как важно учитывать все условия задачи.

Теперь следует вернуться к п. 2.2 и определить получение новой переменной. Очевидно, все её значения описываются рекуррентным выражением (итерационной формулой):

f:=f+1. (35.1)

Очевидно также, что вычисление нового значения f по (35.1) следует выполнять при каждом выявлении в массиве A очередного положительного элемента.

Значение f по завершении алгоритма будет указывать число положительных элементов в A (и одновременно номер последнего элемента B, содержащего результат).

5.1. Порядок решения задачи (корректный).

этап 1: A[1]>0? : да – f:=f+1, B[f]:=A[1]; нет – переходим к следующему этапу.

этап 2: A[2]>0? : да – f:=f+1, B[f]:=A[2]; нет – переходим к следующему этапу.

5.2. Обобщение решения.

этап i: A[i]>0? : да – f:=f+1, B[f]:=A[i]; нет – переходим к следующему этапу.

5.3. Операции последнего этапа соответствуют обобщённым.

5.4. Операции первого этапа уже приведены к общему виду добавлением формулы (35.1); из неё определяется начальное значение f=0.

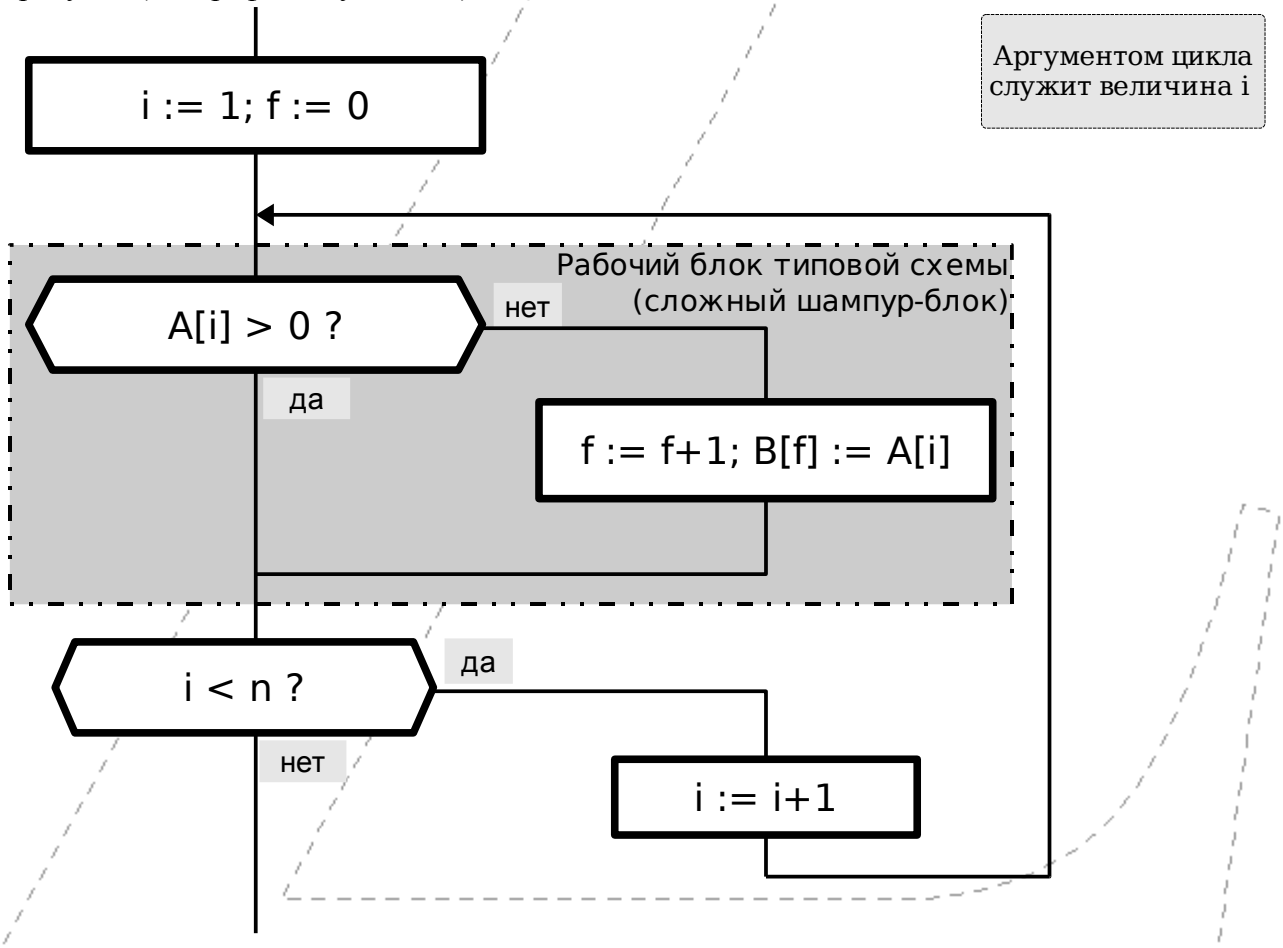
6. Окончательно формируем содержание икон для циклического алгоритма.

6.1. Переменные и выражения для них уже выделены в ходе построения корректного алгоритма, за исключением очевидной итерационной формулы вычисления индекса массива A:

i:=i+1 |i=1...n. (35.2)

6.2. Из (35.2) вытекает условие ОПЦ: i<n.

7. Изображаем типовую схему шампур-блока и заполняем её иконы. Результат показан на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения задачи 5

Как и в задаче 3, имеем сложную (ветвящуюся) структуру рабочего блока циклического визуала.

В данной задаче встретился частный случай для п. 5.

Составление алгоритмов произвольного вида

Задача 6

Дано: матрица A(n,n).

Надо: значения элементов матрицы, лежащих на главной диагонали и правее её, сдвинуть на одну позицию вправо. Элементы последнего столбца запомнить, а элементы главной диагонали принять равными нулю.

Решение.

1. Принимаем формулировку условия с уточнением: результатом запоминания является отдельная величина.

2. Описываем объекты данных.

2.1. К исходным данным относятся:

n – размерность матрицы;

A[1…n,1…n] – двумерный массив, содержащий элементы матрицы.

2.2. К результатам относятся:

A[1…n,1…n] – преобразованная матрица;

B[1…n], причём B[i]=A[n,i], т.е. элементы массива B д.б. равны элементам последнего столбца матрицы A.

3. Метод: последовательно преобразуем строки матрицы, начиная с первой.

После данного пункта методики различаются и нужно выбрать, какой пользоваться; выбор основан на представлении автора, сложна решаемая задача или нет (тогда её метод можно описать одноциклическим алгоритмом). В общем случае для принятия окончательного решения нужно выполнить п. 4; после этого для простой задачи следует возврат к п.3 соответствующей методики.

4. Анализируем порядок решения задачи. В результате убеждаемся, что этот процесс распадается на ряд этапов, на каждом из которых выполняется аналогичный набор операций (обработка элементов одной строки). Однако обработка n-ной строки отличается от обработки предыдущих строк, т.к. в n-ной строке отсутствует сдвиг элементов внутри матрицы (поскольку нет элементов, лежащих правее главной диагонали).

Т.о., во всей задаче можно выделить две крупные однократно выполняемые операции, т.е. две подзадачи первого уровня детализации:

1) Обработка строк с 1-й по (n-1)-ю.

2) Обработка n-ной строки.

Отсюда вытекает, что задача сложная, поэтому продолжаем выполнение данной методики.

5. Выделяем наиболее крупную многократно выполняемую операцию.

5.1. Расписываем начальные этапы.

этап 1: B[1]:=A[1,n]; сдвинуть в 1-й строке элементы A[1,1]...A[1,(n-1)] на места элементов A[1,2]...A[1,n]; A[1,1]:=0.

этап 2: B[2]:=A[2,n]; сдвинуть во 2-й строке элементы A[2,2]...A[2,(n-1)] на места элементов A[2,3]...A[2,n]; A[2,2]:=0.

5.2. Составляем обобщённое описание.

этап i: B[i]:=A[i,n]; сдвинуть в i-той строке элементы A[i,i]...A[i,(n-1)] на места элементов A[i,(i+1)]...A[i,n]; A[i,i]:=0.

5.3. Операции последнего этапа соответствуют обобщённым; на этом этапе обрабатывается (n-1)-я строка матрицы A.

5.4. На каждом этапе также определяется новый индекс из итерационной формулы:

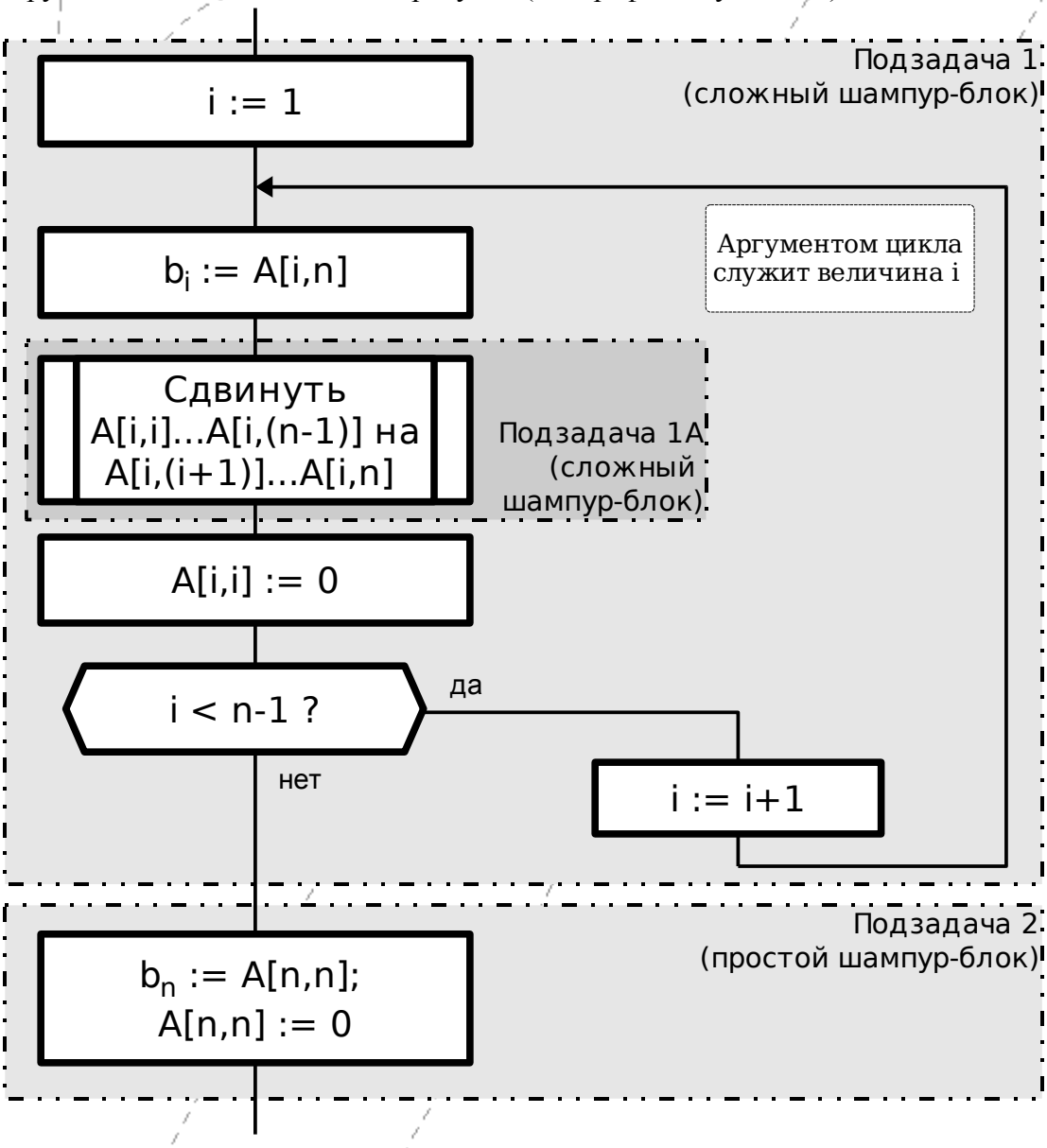
$$i:=i+1 \quad |i=1...n-1. \qquad (36.1)$$

6. Составляем укрупнённую схему алгоритма решения всей задачи.

6.1. Выражения для исходных данных и результатов, множества которых заданы указанием граничных значений индексов, уже составлены в п. 5.

6.2. Из (36.1) вытекает условие ОПЦ: i<n-1.

Укрупнённая схема показана на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения задачи 1

7. Дальнейшей детализации подлежит операция сдвига элементов i-той строки. Рассматриваем её как подзадачу второго уровня детализации, обозначив буквенным индексом «А».

Дано: матрица A(n,n).

Надо: сдвинуть элементы A[i,i]...A[i,(n-1)] на места элементов A[i,(i+1)]...A[i,n].

Решение. (к номерам пунктов методики будем добавлять приставку-индекс подзадачи)

А2. Описываем объекты данных.

А2.1. К исходным данным относятся:

n – размерность матрицы;

A[i,i]...A[i,(n-1)] – массив, содержащий элементы i-той строки матрицы, подлежащие сдвигу.

А2.2. К результатам относятся:

A[i,(i+1)]...A[i,n] – преобразованная часть строки матрицы.

А3. Метод: последовательно переносим вправо каждый элемент, начиная с последнего.

А5. Наиболее крупная операция – перенос одного элемента – повторяется многократно; следовательно, мы пришли к простой задаче и строим визуал с одним циклом.

Порядок решения задачи (для краткости без выделения пунктов):

этап 1: A[i,n]:=A[i,(n-1)].

этап 2: A[i,(n-1)]:=A[i,(n-2)].

этап j (т.к. имя i уже занято): A[i,(n-j+1)]:=A[i,(n-j)].

этап r (последний): A[i,(i+1)]:=A[i,i].

Все этапы изначально записаны в единообразном виде; нужно лишь вывести выражение для индекса j:

$$j:=j+1 \quad |j=1...n-i.$$

Теперь имеем все рабочие формулы.

А6. Окончательно формируем содержание икон для циклического алгоритма.

А6.1. Переменные и выражения для них уже имеются.

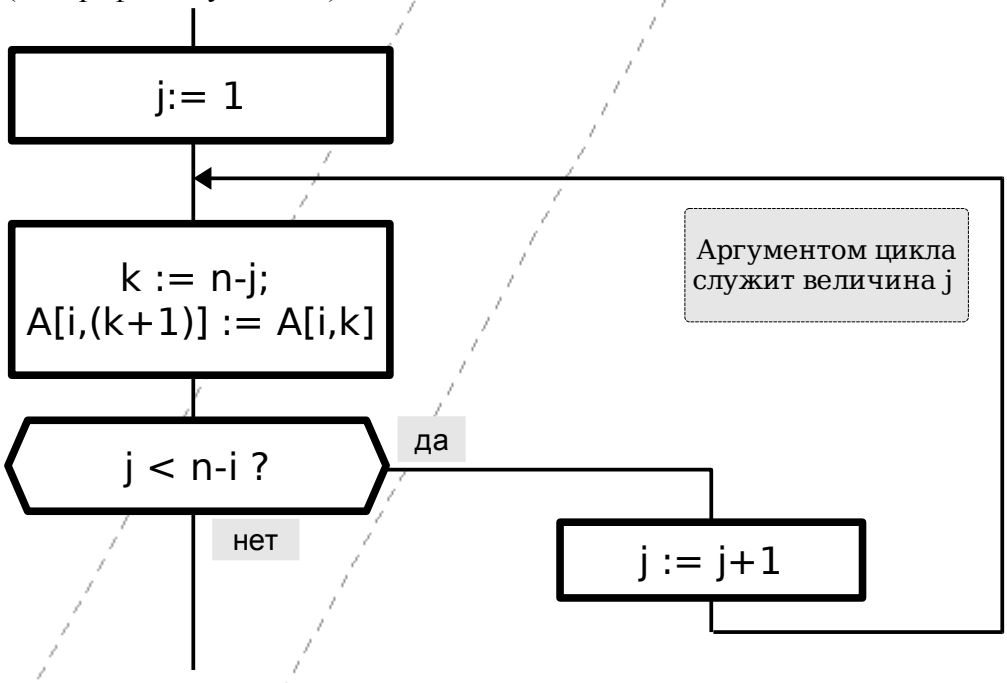
С целью упрощения выражений для индексов матрицы введем еще одну промежуточную величину k:=n-j. Тогда обобщённая операция i-того этапа запишется в виде:

$$A[i,(k+1)]:=A[i,k].$$

А6.2. Сравнивая выражения для 2-го индекса результата операции на j-том и r-том этапах, получаем условие ОПЦ:

$$n-j+1=i+1 \text{ или } j=n-i.$$

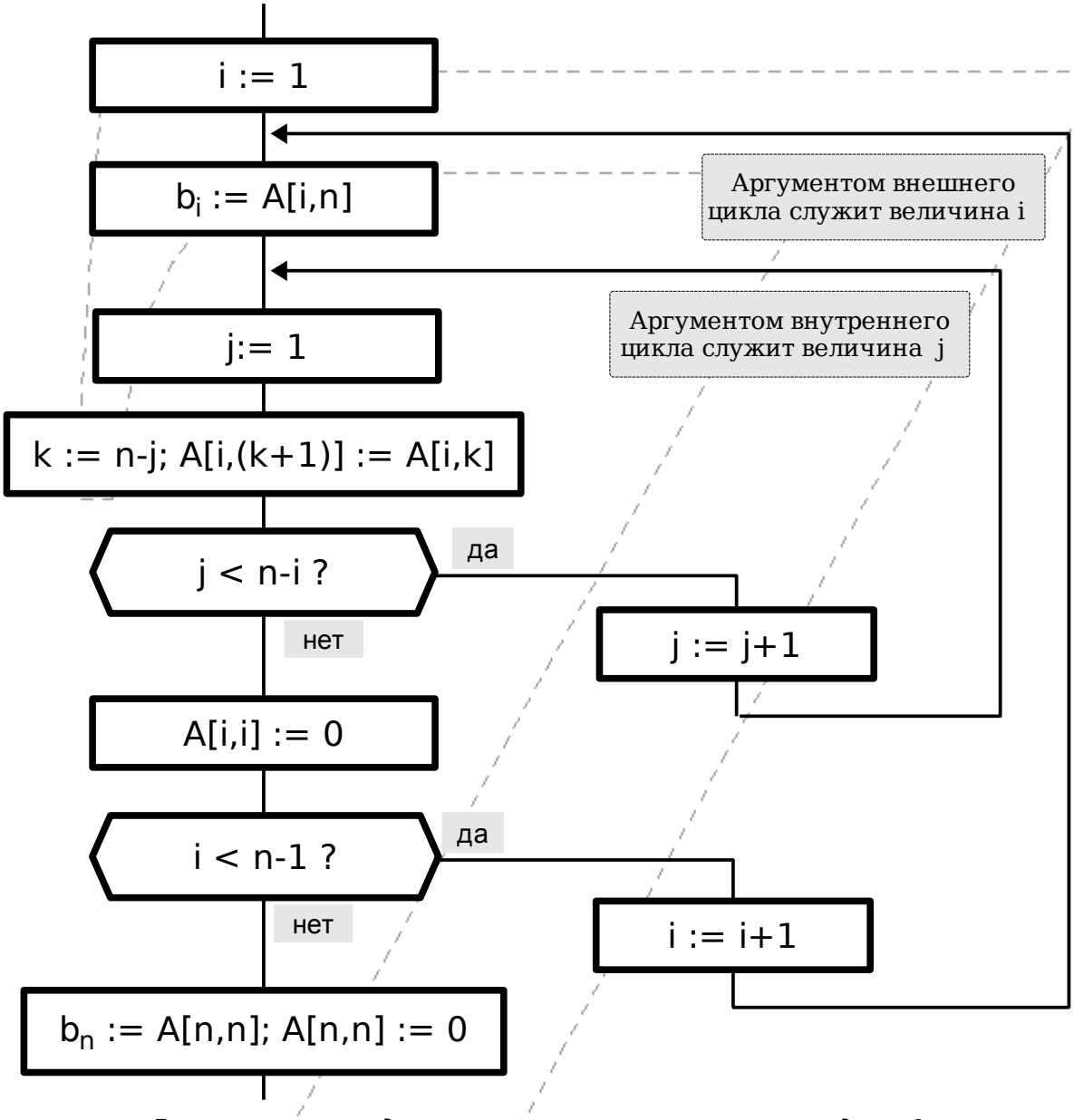
А7. Изображаем типовую схему шампур-блока и заполняем её иконы. Результат показан на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения подзадачи 1А

На этом визуализация подзадачи завершена; поскольку других подзадач нет (что очевидно из укрупнённой схемы), то завершен и вывод текстов подробной схемы для всей задачи.

Составляем подробную схему всей задачи, подставляя подробную схему подзадачи "А" вместо укрупнённого блока; результат показан на рисунке (см. графическую часть).



Визуализация содержания алгоритма решения задачи 6

Если подробная схема чересчур сложна, то визуалы подзадач оформляют как вставки.

---конец сводной части [N]---

Б. Основной материал по <наименование темы>

---содержание сводной части [X] включается в текст Приложения 8 согласно первого по порядку заголовка внутри части. Переменный индекс [X] заменяется на литеру надзаголовка для данной задачи (по порядку следования литерованных частей в тексте согласно плану приложения).

Приложение 8 по прежней нумерации являлось Приложением 11---

Приложение 8.

КОГНИТИВНОЕ ОБЕСПЕЧЕНИЕ ФОРМАЛИЗАЦИИ ЗНАНИЙ

В приложении кратко описаны элементы методического и отчасти математического обеспечения (авто)формализации профессиональных знаний до уровня постановок задач, включающих определение объектов и алгоритмов решения. Включены элементы алгоритмизации и программирования для информашин.

Приложение рассчитано на самостоятельное практическое использование, поэтому в нем повторяются некоторые положения курса лекций.

<...>

---начало сводной части [X]---

[X]. Элементы методического обеспечения алгоритмики

[X]1. Методики формирования решения задачи

Далее (с новой страницы) приведена методика.

МЕТОДИКА ВИЗУАЛИЗАЦИИ ПРОЦЕССОВ ПЕРЕРАБОТКИ ДАННЫХ

Обозначения и определения

1. Назначение методики – визуализация процесса получения результата задачи.

В основе лежит методика составления алгоритмов по В. Ляховичу.

2. Объект методики (исходное описание) – неалгоритмическое словесно-формульное.

3. Версия техноязыка – ДРАКОН-1 со следующими требованиями к подъязыкам.

3.1. В маршрутном подъязыке возможны следующие типы структур алгоритмов (дракон-схемы) по уровню обобщения.

- *Подробные* схемы описывают процесс решения задачи в терминах используемого языка программирования X со степенью подробности, необходимой в указанном языке.
- *Укрупнённые* схемы содержат *укрупнённые блоки*, реализующие операции произвольного вида; схема в целом имеет один вход и один выход, т.е. является шампур-блоком.

Блок подробной схемы – икона «Действие», «Вопрос» либо макроикона «Переключатель».

Укрупнённый блок м.б. шампур-блоком либо развилкой (переключателем). Каждый укрупнённый шампур-блок описывается самостоятельной подробной схемой.

3.2. В командном подъязыке (текстовой части икон) приняты следующие соглашения.

3.2.1. Язык текста укрупнённых схем м.б. любым, в т.ч. естественным.

3.2.2. Текст подробных схем удовлетворяет следующим формальным требованиям.

3.2.2.1. В иконах «Действие» допускается использовать выражения вида:

$$X := A, \qquad (1)$$

где X (A) – любая переменная (арифметическое выражение).

$$Y_i := f(Y_{i-1}), \qquad (2)$$

где Y – одна и та же величина, принимающая ряд значений i=1...n, в частности текущее (i) и предшествующее (i-1).

Выражение (1) выполняется так же, как оператор присваивания в ЯП. Выражение (2) называется *рекуррентным*.

3.2.2.2. В иконах «Вопрос» или «Вариант» допускается использовать только логические выражения, в частности отношения, например X<Y+2.

3.3. Объекты (данные) задачи декларируются как величины следующих типов:

- *скалярный* (по Ляховичу - переменная) – значение, обозначенное именем (меткой);
- *массив* – множество значений (элементов), обозначенных одним именем с различными целочисленными индексами значений, изменяющимися по порядку.

Имя (метка) величины образуется по общеязыковым правилам (см. п/р 1.1 Приложения 1).

4. Результат методики – содержимое (тело) визуала обработки данных в форме шампур-блока одного из следующих видов.

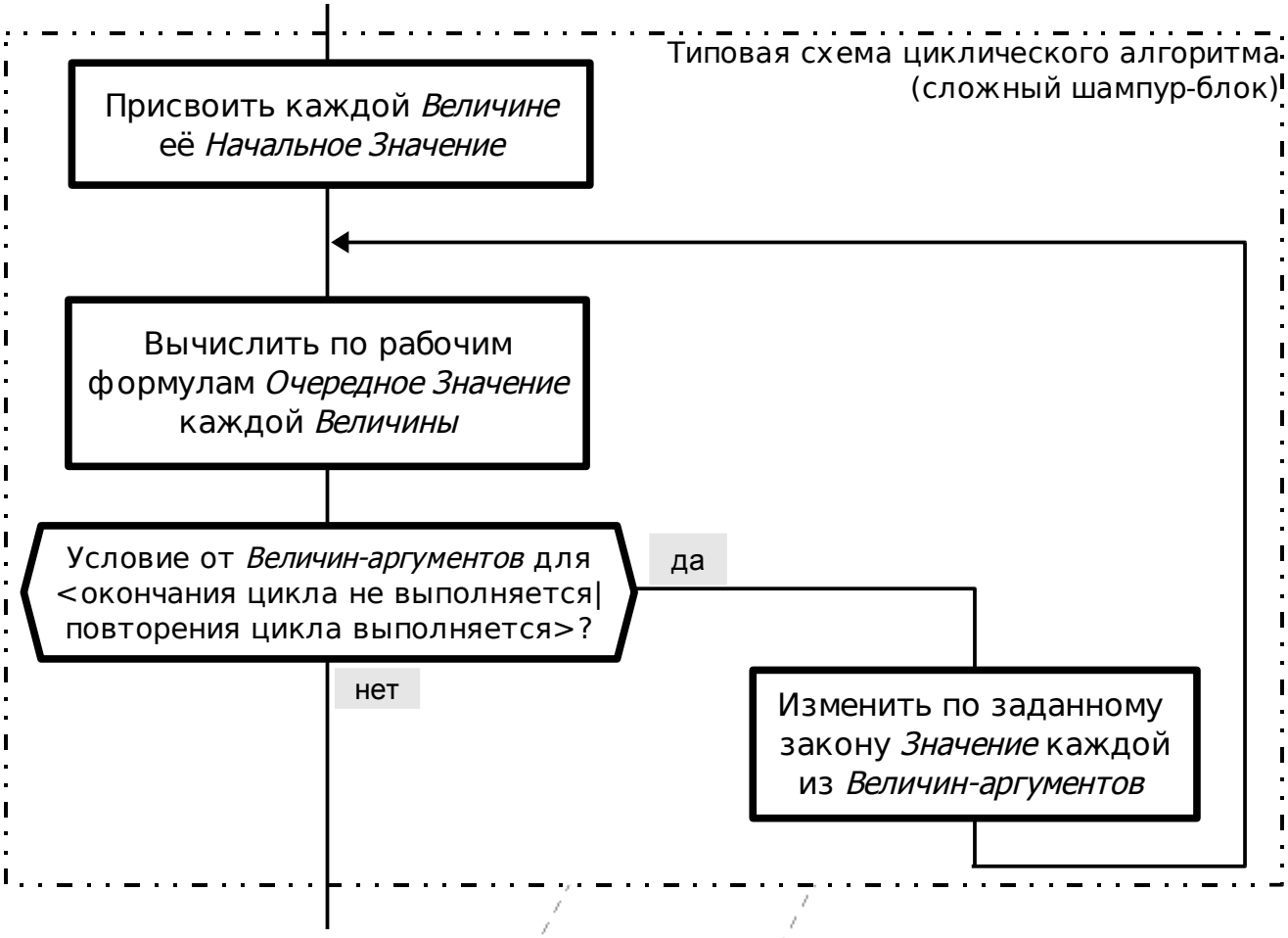
- *Циклический* – в котором получение результата обеспечивается многократным вычислением по одним и тем же формулам при разных значениях входящих в них переменных.
- *Произвольный* – в котором результат получается в ходе процесса более сложной структуры.

Произвольный алгоритм м.б. линейным или разветвляющимся, причём отдельные линейные участки могут включать циклические алгоритмы.

5. Базовая метаструктура визуализации определяется в зависимости от формы визуала.

5.1. Циклический алгоритм визуализирует т.н. *типовая схема* (укрупнённая), на техноязыке имеющая вид шампур-блока на базе гибридного цикла (см. графическую часть)

Для визуализации циклического алгоритма необходимо располагать т.н. *рабочими формулами*, т.е. такими, многократное вычисление по которым дает искомый результат.



Шампур-блок (типовая схема) тела циклического алгоритма

Обычно рабочая формула – это рекуррентное выражение вида (2); для его вычисления необходимо располагать также начальным значением Y_1.

5.2. Произвольный алгоритм визуализируется схемой, содержащей в качестве укрупнённых блоков любое сочетание шампур-блоков и/или развилок согласно правилам техноязыка, причём любой шампур-блок может содержать один и более циклов; каждый цикл описывается типовой схемой. Цикл, который составляет содержание рабочего блока (или части рабочего блока) другого цикла, является *вложенным*.

Типовая схема выбрана как оптимальная для визуализации произвольных алгоритмов, содержащих вложенные циклы.

6. Основная идея методики в терминах ТФЗ-ДРАКОН – эскизная визуализация алгоритма получения результата задачи как укрупнённой дракон-схемы, где укрупнённый блок обобщённо описывает свою часть, а текст икон записывается на неформальном языке. Далее от укрупнённой схемы переходят к подробной путем детализации укрупнённых блоков (итеративной для произвольных алгоритмов) и одновременно от командного языка укрупнённых схем – к командному языку подробных схем как совокупности выражений (1) и (2) над величинами.

В заключение все полученные тексты размещаются в блоках подробной схемы. Получается уточнённая (формальная) спецификация обработки данных задачи на гибридном техноязыке ДРАКОН-Х, т.е. дракон-программа получения результата.

Алгоритм каждого вида составляется с применением своего варианта методики (см. далее); оба варианта используют единую нумерацию шагов, отличаясь между собой их составом (поэтому некоторые номера м.б. пропущены) и содержанием некоторых однономерных шагов.

7. Для понимания дальнейшего существенны также следующие допущения Ляховича.

7.1. Все задачи для алгоритмизации неявно разделяются на два вида: *простыми* считаются те, решение которых можно описать циклическим алгоритмом; все остальные относятся к

сложным. В общем случае вид задачи заранее неизвестен и определяется исполнителем методики интуитивно (на базе опыта алгоритмизации, накопленного в т.ч. путем разбора предлагаемых автором примеров).

7.2. Должен быть известен математический метод решения алгоритмизуемой задачи. Если это не так, метод д.б. найден не позднее шага 3 методики, возможно, эмпирически .

7.3. Детализация понимается более узко, чем в ТФЗ-ДРАКОН, а именно только как раскрытие содержания блоков укрупнённой схемы уже заданной структуры до блоков (подсхем) подробной схемы. Поэтому если при визуализации произвольного алгоритма заданная структура укрупнённой схемы оказалась неоптимальной, методика применяется повторно сначала с выбором новой структуры.

7.4. Методика составлялась до появления техноязыка, и диоформой всего визуала в ней неявно считается примитив (т.е. полученный шампур-блок подставляется в заготовку-примитив для завершения визуализации).

7.5. Методика охватывает только процесс преобразования исходных величин в результатные; предполагается, что и те и другие находятся в памяти исполнителя визуала.

Составление алгоритмов с одним циклом

Общие положения

1. Для однозначного задания циклического алгоритма необходимо указать три объекта:

- множество начальных значений переменных (точнее, выражений их присвоения);
- множество рабочих формул;
- условие окончания повторения цикла (ОПЦ) или повторения цикла (ПЦ).

2. Визуализация алгоритма в общем происходит следующим образом.

2.1. Если все объекты заданы в явном виде, то для визуализации алгоритма достаточно разместить их в типовой схеме шампур-блока.

2.2. Если объекты не заданы в явном виде, то предварительно требуется поэтапное описание процесса решения задачи и вывода на его основе рабочих формул для вычисления результатов. Из этого же описания и полученных формул выводятся начальные значения всех переменных. Справедливость полученных формул можно далее доказать методом математической индукции.

Условие ОПЦ (ПЦ) выводится затем из сравнения выражений для некоторой величины на i-том и последнем этапах.

3. Заполнив объектами типовую схему алгоритма, мы получаем подробную схему; если была доказана справедливость формул, то считается доказанной и правильность схемы.

Содержание методики

1. Формулируется условие задачи.

2. Определяются состав и структура объектов данных, т.е. совокупность скалярных величин и массивов, которыми исходные данные будут представлены в задаче.

2.1. Выделяются исходные данные:

⇒ определяется, какие величины отнести к исходным;

⇒ выбирается, должна быть величина скалярной или массивом;

⇒ каждой величине назначается уникальное осмысленное имя.

В простых задачах используется правило выделения и выбора типа величины: если дано множество значений некоторой величины, закон изменения которой отсутствует, то множество рассматриваем как массив и включаем в исходные данные. В противном случае (закон изменения известен) массив можно не создавать, а использовать рекуррентное выражение; при исполнении визуала оно потребует меньше памяти.

2.2. Выделяем результаты, для чего:

⇒ определяем количество результатных величин;

⇒ назначаем каждой из них имя;

⇒ определяем, должна ли та или иная величина быть массивом.

3. Выбираем метод решения задачи и разбиваем процесс её решения на этапы с равным числом аналогичных операций.

Проверяем, есть ли для вычисления всех результатов формулы с фиксированным числом операций каждая. Если есть, то переходим к п. 6.

5. Выводятся рабочие формулы, для чего:

5.1. Записываются операции 1-го, 2-го, 3-го этапов решения задачи, используя выражения вида (1) и (2). Результат выполнения одной и той же операции на разных этапах обозначается одним именем с индексом, равным номеру этапа;

5.2. Записываются операции, выполняемые на i-том этапе, как результат обобщения записей для этапов 1...3, т.е. выводятся рабочие формулы для вычисления всех переменных. При этом каждая из этих величин выражается в виде функции либо от номера этапа, либо от предшествующих значений той же величины (итерационная формула). Если данный пункт не может быть выполнен для всех величин[7], то применить методику составления алгоритмов произвольного вида (см. далее);

5.3. Записываются операции последнего этапа, если они известны;

5.4. Все операции, выполняемые на каждом из рассмотренных этапов (обычно операции 1-го и последнего этапов) приводятся к виду формул i-того этапа. При этом выявляются начальные значения переменных.

Для частного случая алгоритмизации, когда некоторая переменная алгоритма зависит от её предшествующего значения, методика уточняется следующим образом.

Если при выполнении п. 5 обнаруживается, что переменная зависит от своего предшествующего значения (на предыдущем этапе решения задачи), то:

⇒ обозначаем эту переменную именем (вернувшись к п. 2);

⇒ выводим для неё рекуррентную формулу;

⇒ определяем положение этой формулы в описываемом процессе решения задачи (перейдя к п. 5).

Далее действуем в соответствии с методикой.

Этот же подход можно использовать и в том случае, когда переменная зависит от номера этапа, но эта зависимость сложная.

6. Формируем содержание икон типовой схемы алгоритма:

6.1. Для каждой переменной, включённой в формулы i-того этапа, выписываем начальное и конечное значения и закон изменения. Для каждого массива вводим служебную переменную, содержащую значения его индексов.

6.2. Выявляем условие окончания (повторения) цикла. Оно может:

- Быть известно из условия задачи или метода её решения.
- Определяться предельным значением некоторой переменной, полученным в п. 6.1, т.е. имеет вид отношения, связывающего текущее и предельное значения этой переменной.
- Быть получено, если известны операции последнего этапа: приравниваем выражения для некоторой переменной на i-том и последнем этапах.

[7] Это означает, что циклический алгоритм для всей задачи построить невозможно и следует строить алгоритм другого вида либо представить алгоритм как совокупность подалгоритмов циклического и/или прочих видов.

7. Изображается типовая схема (шампур-блок) циклического алгоритма и заполняется каждая икона в соответствии с её назначением. При этом в рабочей иконе записываются все выражения, полученные на i-том этапе либо заданные рабочие формулы для вычисления результатов; если выражения описывают процесс сложной структуры, то рабочая икона детализируется в соответствующую ей схему (шампур-блок). При заполнении схемы индексы сохраняются лишь для величин, рассматриваемых как массив; во всяком случае, если значение величины используется только в пределах одного этапа – того же, на котором оно вычисляется или следующего, – то рассматривать эту величину как массив нет необходимости.

Составление алгоритмов произвольного вида

Общие положения

1. Методика используется для алгоритмизации сложных задач.

Для визуализации алгоритма произвольного вида необходимо представлять структуру путей к решению задачи в общем виде (как укрупнённую схему) и располагать выражениями для укрупнённых блоков.

2. Смысл данной методики в том, что вначале составляется укрупнённая линейная, разветвляющаяся или циклическая (с одним циклом) схема алгоритма. Затем составляются подсхемы отдельных укрупнённых блоков (независимо друг от друга). Наконец, составляется подробная схема алгоритма решения всей задачи механическим объединением подсхем в соответствии с укрупнённой схемой.

При таком подходе процесс составления алгоритма предполагает несколько последовательных шагов, на каждом из которых составляется элементарная по своей структуре схема. Сложность всей задачи сказывается лишь на числе шагов.

Структура укрупнённой схемы (подсхемы) никак не ограничивается, и в общем случае при её визуализации как примитива возможны пересечения линий; в этом случае необходим топологический переход к силуэту.

3. Данная методика выполняется рекурсивно (т.е. вызывает сама себя), пока весь алгоритм не будет сведен к схемам элементарной структуры (подробным), а также может использовать ранее определённую методику составления алгоритмов с одним циклом.

4. Понятия «наиболее крупная операция» и «наиболее крупный этап» нестроги и применяются к конкретной задаче эвристически, на основе здравого смысла и опыта работы по данной методике.

Содержание методики

1. Чётко формулируется условие задачи.

2. Определяются состав и структура объектов данных задачи (исходных и результатных), как на аналогичном шаге предыдущей методики.

3. Формулируется метод решения задачи в наиболее общем виде (без лишних подробностей).

4. Выделяются наиболее крупные этапы решения задачи, которые выполняются однократно и охватывают в совокупности процесс решения всей задачи. При отсутствии таких этапов следует переход к п.5.

4.1. Если такие этапы есть (в сложной задаче их не менее двух), то:

⇒ последовательно описывается каждый этап;

⇒ составляется укрупнённая линейная или разветвляющаяся схема алгоритма всей задачи с одним выходом (шампур-блок);

⇒ следует переход к п.7.

5. Выделяется наиболее крупная операция, которая выполняется многократно для решения всей задачи.

6. Составляется укрупнённая схема алгоритма с одним циклом по ранее определённой методике. При этом, описывая процесс решения задачи в терминах операции п.5, на каждом из этапов (1, 2, 3,..., i) указываем множества исходных данных и результатов заданием, в общем случае, их граничных значений (или граничных значений индексов).

7. Для каждого укрупнённого блока полученной схемы составляем свою схему алгоритма, рассматривая поочередно каждый блок как самостоятельную задачу и применяя к нему настоящую методику, начиная с п.1. Очередность рассмотрения блоков в одной укрупнённой схеме – от входа к выходу.

---конец сводной части [X]---