МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
АГРАРНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ИНФОРМАТИКА

Методические указания по изучению дисциплины «Информатика»
и задания для выполнения контрольной работы
студентам заочной формы обучения всех специальностей

Минск 2007

УДК 004(07)
ББК 32.81я7
В

Рекомендовано научно-методическим советом агроэнергетического факультета БГАТУ

Протокол № 10 от 15 мая 2007 г.

Составители: д-р техн. наук, доцент *М.А. Прищепов*,
канд. техн. наук, доцент *Н.В. Исаеня*,
ст. преподаватель *Е.В. Севернёва*,
ст. преподаватель *Н.М. Жалобкевич*

Рецензент: канд. техн. наук**,** доцент *Ю.Н. Силкович*

Вычислительная техника и информатика : метод. указания по изучению дисциплины «Информатика» и задания для выполнения
В контрольной работы студентам заочной формы обучения всех специальностей / сост. М.А. Прищепов, Н.В. Исаеня, Е.В. Севернёва, Н.М. Жалобкевич. — Минск : БГАТУ, 2007. — 56 с.

УДК 004(07)
ББК 32.81я7

ОБЩИЕ МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ «ИНФОРМАТИКА»

Дисциплина «Информатика» призвана обеспечить компьютерную подготовку студентов при решении научно-технических задач. Целью данной дисциплины являются:

– формирование необходимых знаний и навыков использования современных базовых компьютерных технологий в качестве инструмента для решения практических задач в своей предметной области;

– развитие логико-алгоритмического мышления у студентов, позволяющего осуществлять математическую формулировку различного ряда прикладных задач с дальнейшим их решением на персональном компьютере (ПК).

В результате изучения дисциплины студенты должны знать:

- устройство персонального компьютера и основные принципы обработки информации;
- виды программного обеспечения ПК и их функциональное назначение;
- структуру и основные возможности прикладного обеспечения;
- сущность математической формулировки прикладных задач и численных методов их решения.

Студенты должны уметь:

- работать с операционными системами, пользоваться прикладным программным обеспечением при решении поставленных задач;
- осуществлять математическую формулировку прикладных задач и реализовывать её в виде алгоритмов и программ на алгоритмическом языке;
- использовать численные методы для решения задач.

Поставленная цель должна быть достигнута в течение 2-х семестров. В первом семестре на обзорных лекциях и лабораторных занятиях студенты знакомятся с устройством ПК, операционной системой, прикладным программным обеспечением и получают необходимые навыки по работе с ним, а в конце семестра сдают зачёт. Кроме того, в первом семестре на обзорных лекциях они знакомятся с вопросами математической формулировки при-

кладных задач, составления алгоритмов и программ, получают задание на контрольную работу, которую выполняют в течение второго семестра.

В следующем семестре на обзорных лекциях изучают дополнительный теоретический материал и закрепляют полученные знания на лабораторных занятиях, а в конце его сдают экзамен.

Таблица 1

Темы обзорных лекций

Тема	Содержание	Кол-во часов
1 Базовая аппаратная конфигурация ПК. Операционные системы. Прикладное программное обеспечение.	Принцип обработки информации в ПК. Назначение и характеристики его основных устройств. Виды операционных систем и их функциональное назначение. Операционная система Windows. Графический интерфейс Windows и его основные элементы. Текстовый редактор Word. Общая характеристика. Интерфейс. Панели инструментов. Табличный процессор Excel и его интерфейс. Работа с таблицами средствами Excel.	2
2 Математическая формулировка прикладных задач. Составление алгоритмов и программ на алгоритмическом языке.	Понятие алгоритма. Разновидности структур алгоритмов. Типовые приёмы алгоритмизации. Составление алгоритмов и программ разветвляющейся и циклической структуры.	4
3 Решение задач на ПК численными методами.	Численные методы решения нелинейных уравнений, вычисление интегралов.	4

Таблица 2

Темы лабораторных работ

Тема	Содержание	Кол-во часов
1 Работа в операционной системе Windows.	Главное меню, его назначение и настройка. Работа с окнами. Работа с дисками, папками, файлами. Копирование, удаление, просмотр файлов.	2
2 Работа с текстовым редактором Word.	Создание текстового документа и сохранение его на диске. Работа с фрагментами текста (копирование, удаление, просмотр, изменение шрифтов, цвета). Создание таблиц средствами Word. Набор и редактирование формул. Работа с графическими объектами.	4
3 Работа с табличным процессором Excel	Интерфейс Microsoft Excel. Структура окна. Создание таблиц и сохранение их на диске. Вычисления в электронных таблицах. Построение диаграмм.	2
4 Решение прикладных задач средствами алгоритмического языка	Математическая формулировка прикладных задач. Составление алгоритмов и программ их решения с последующим получением результата на ПК.	4
5 Решение нелинейных уравнений	Графический и программный способы отделения корней. Уточнение корней нелинейных уравнений. Решение нелинейных уравнений на ПК.	2
Вычисление интегралов	Численные методы вычисления интегралов. Вычисление интегралов на ПК.	2

Примечание – При проведении лабораторных работ в первом семестре используются обучающие программы по Windows, Word, Excel.

ЗАДАНИЕ 1

1 Понятие, свойства и способы описания алгоритма

Под алгоритмом понимается «точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату» (ГОСТ 19.781–74).

Алгоритм включает систему правил, определяющих содержание и конечную последовательность действий (шагов и операций), выполняемых над некоторыми объектами с целью переработки исходных и промежуточных данных в искомый результат. Это предписание конкретному исполнителю о

том, какие действия, над какими объектами и в каком порядке следует выполнять для решения поставленной задачи.

При разработке алгоритмов следует учитывать ряд требований, совокупность которых формирует его свойства*:* ***определенность, дискретность, конечность, результативность, рациональность, массовость.***

Указания, составляющие алгоритм, должны быть четкими и однозначными, не допускать произвольного или двоякого толкования. Это свойство называют *определенностью.* Оно является гарантией того, что алгоритм может быть выполнен объектами, не обладающими интеллектуальными способностями, в частности персональным компьютером (ПК). При составлении алгоритма нельзя рассчитывать на профессионального исполнителя, который может проанализировать и как-то исправить ход решения задачи в случае необходимости.

Предопределенный алгоритмом вычислительный процесс может быть разделен на отдельные этапы (шаги), представляющие собой элементарные операции. В результате чего возникает упорядоченная запись совокупности предписаний (директив, команд), образующих непрерывную структуру алгоритма. В этой упорядоченной записи выполнение действий очередного предписания допустимо лишь после исполнения предыдущего. Возможностью поэтапной детализации алгоритма путем разложения сложной структуры на ряд простых, строго очерченных действий определяет свойство алгоритма, называемое *дискретностью.* Отметим также, что в алгоритмах предписывается обработка дискретных значений параметров объектов, принимающих в любой момент времени конкретные цифровые (символьные или логические) значения.

Вычислительный процесс после выполнения заданной алгоритмом конечной последовательности действий должен заканчиваться выдачей результатов или сообщением о невозможности решить задачу. Эти взаимосвязанные свойства алгоритма называются *конечностью* и *результативностью.*

Разработанные алгоритмы могут быть представлены на физическом носителе информации различными способами, наиболее известными из которых являются: словесный (средствами языка человеческого общения с тщательно отобранным набором слов и фраз), структурно-стилизованный (языком псевдокодов), графический (схемами из графических блок-символов) или программный (текстами программ).

Наиболее распространенным способом представления алгоритма является ***графический.*** В графическом представлении алгоритмы изображаются в виде блок-схемы, дополненной элементами словесной или математической записи. Схема алгоритма включает геометрические фигуры (блочные символы), соединенные между собой стрелками (линиями), указывающими порядок выполнения операций. Блочные символы стандартизированы и различаются по типу выполняемых действий (ГОСТ 19.701–90, международные стандарты ИСО 5807–85).

В схеме начало и завершение алгоритма, а также вход и выход из вспомогательных алгоритмов отмечаются соответственно блочными символами «начало» и «конец» (рисунок 1, *а, б*), блоки 1 и 2). Эти блоки, в отличие от большинства других, используются по одному в алгоритме и отмечают как бы начало и конец пути обработки информации. Каждая схема обязательно должна начинаться и заканчиваться этими символами.

Изображенные на рисунок 1 *в, г*) блочные символы в виде параллелограмма (блоки 3 и 4) используют для обозначения операций ввода; вывода данных.

Блок, отражающий вычислительный процесс, применяют для обозначения одной или группы операций, изменяющих значение, форму представления или размещения данных (рисунок 1 *д,* блок 5*)*. Производимые операции в этом блоке записывают в любой форме с использованием математических формул, выражений и пояснений на естественном языке.

Логический блочный символ «решение» (рисунок 1 *е–и*), блоки 6, 7, 8, 9 соответственно) используют для обозначения выбора направления алгоритма

в зависимости от некоторого условия (условий). В блоке указывают условие, вопрос или решение, определяющие дальнейшее направление выполнения алгоритма. Условия могут быть простыми (рисунок 1 *е,* блок 6) и составными (рисунок 1 *з,* блок 8). В них должны быть учтены абсолютно все возможные варианты следования процесса при решении задачи.

Из блока проверки условия может выходить два, три и более (блок 9) информационных потока, что отличает его от других блочных символов имеющих не более одного выхода. Выходящие из блока линии должны снабжаться пояснениями о направлениях исполнения алгоритма при выполнении или невыполнении приведенного условия (например, «да», «нет», «<0», «=0», «>0», «=1»,«+» «–» или др.).

Блочный символ модификации (рисунок 1 *к, л*, блоки 10 и 11) символизируют начало циклических вычислений (заголовок цикла), для управления которыми он используется. Внутри блока указывается переменная цикла и параметры, характеризующие закон ее изменения, например, $I = A_{НАЧ}, A_{КОН}, \Delta A,$ где I – переменная цикла, $A_{НАЧ}$ и $A_{КОН}$ – начальное и конечное значения переменной) цикла, ΔA – шаг ее изменения (переменная цикла изменяется, от $A_{НАЧ}$ до $A_{КОН}$ с шагом ΔA). Если шаг равен 1, то ΔA можно не указывать. Кроме входящей линии блок модификации имеет одну выходящую (на рисунок 1, *л),* обозначенную «Вых»), а также линии, отмечающие передачу вычислительного процесса на обработку для циклических вычислений «Цикл» и возврат в начало для изменения переменной цикла «Изм. пер.».

Для обращения к вычислению по подпрограмме (стандартной или разработанной пользователем) в схеме используют блок-символ «предопределенный процесс», изображенный на рисунок 1 *м)* (блок 12). Он как бы заменяет алгоритм подпрограммы (вспомогательный алгоритм) и указывает, что информационный поток передается подпрограмме. По завершении вычислительного процесса в подпрограмме результаты расчета возвращаются в основной алгоритм, в котором процесс вычислений возобновляется с блока,

следующего за блоком обращения к подпрограмме. Блок «предопределений и процесс» используют при организации вспомогательных алгоритмов, оформленных автономно в виде отдельного модуля, или при обращении к библиотечным подпрограммам. Для уменьшения количества пересечений и длины линий, символизирующих пути следования информационных потоков, допускается их разрывать, вставляя в места разрыва соединители.

Если линия разрывается между блоками, размещёнными на одной странице, то в качестве соединителя используют соответствующие символы (рисунок 1, *н, о*). В блочный символ вписывают номера блоков, в которые вычислительный процесс передаётся или из которых он поступил. Так, соединитель (на рисунке 1 *н)* указывает, что вычислительный процесс передаётся на вход блока 15, а соединитель (на рисунке 1 *о),* что вычислительный процесс поступил из выхода блока 10.

Если же линии соединяют блоки, расположенные на разных страницах, то используется символ межстраничного соединителя, в который вписывают не только номера блоков, но и номера страниц. На рисунке 1 *п* межстраничном соединителе указано, что вычислительный процесс передаётся на вход блока 23, изображённого на странице 10. В этом случае перед входом блока 23 будет стоять (приведённый на рисунке 1 *р)* межстраничный соединитель (информация передана с выхода блока 12, расположенного на странице 6).

Для пояснений особенностей функционирования отдельных блоков или групп блоков, принятых допущений и назначений отдельных элементов, обозначений переменных и др. в схеме алгоритмов могут включаться комментарии (рисунок 1 *с*) .

Схема является самым наглядным и простым способом представления алгоритма. В ней четко прослеживаются порядок выполнения действий, потоки информации и пути их следования, которые отмечаются линиями со стрелками (стрелки допускается опускать, если потоки направлены сверху вниз и слева направо). Линии по отношению к блокам могут быть входящими и выходящим Количество входящих линий для всех блоков не ограниче-

но. Выходящая же линия для большинства блоков может быть только одна (исключение – блоки проверки условия). В схеме блоки, за исключением соединителей, могут нумероваться для простоты дальнейшего описания их работы, организации комментариев и использования соединителей. Номера проставляют в верхней части графического символа в разрыве его начертания, как это сделано на рисунке 1.

Внутри блоков и рядом с ними делаются записи и обозначения, уточняющие выполняемые функции.

Эти записи могут производиться в любой удобной для разработчика форме. Они не имеют каких-либо существенных ограничений (на язык, обозначения, символы и др.), однако должны быть понятны всем, кто будет пользоваться алгоритмом. Единственное ограничение накладывается на последовательность записей – они должны читаться (использоваться при работе алгоритма) слева направо и сверху вниз независимо от направления потоков информации.

Алгоритмы целесообразно разрабатывать поэтапно (по шагам). Сложные задачи следует разбивать на достаточно простые, легко воспринимаемые части.

Логика алгоритма должна опираться на минимальное число достаточно простых управляющих базовых структур. При разработке схем алгоритмов необходимо соблюдать некоторые требования:

- в схеме алгоритма все линии от блока «начало» до блока «конец» не должны иметь разрывов, не помеченных соединителями. Все линии, указывающие последовательность выполнения действий, должны быть замкнутыми;
- в схеме должны четко прослеживаться потоки информации. Блоки следует размещать таким образом, чтобы избегать пересечения линий. При передаче управления в схеме «снизу-вверх» или «справа налево» линии обязательно помечают стрелками;
- не допускается передача управления в никуда. «Источник» передачи управления и «получатель» должны быть четко обозначены.

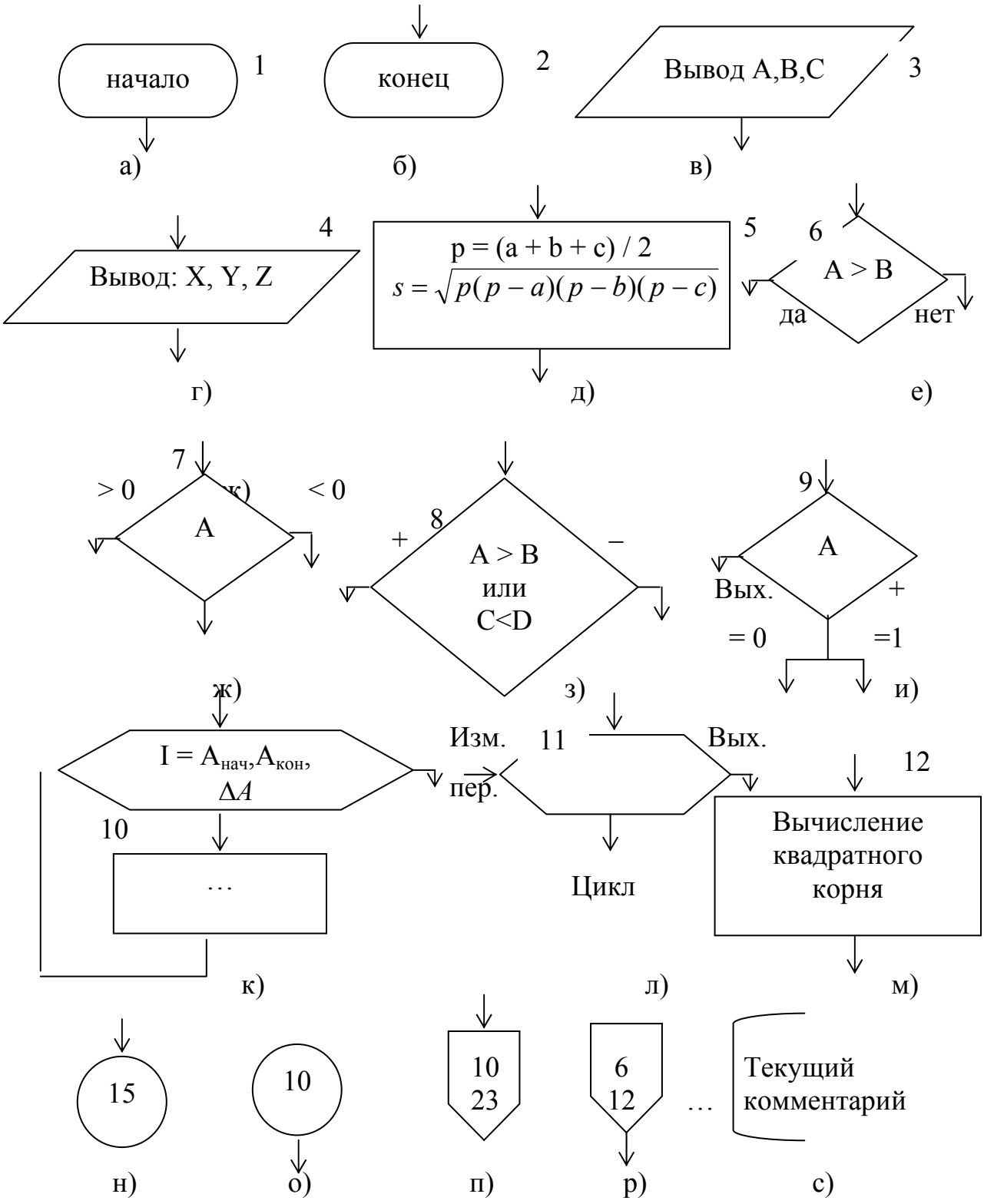

Рисунок 1 Наиболее употребляемые в схемах алгоритмов блоки-символы

2 Основы программирования на языке Турбо Бейсик (Turbo Basic – TB)

2.1 Классификация данных

Алфавит языка включает прописные и строчные буквы латинского алфавита *(A... Z, a…z)* и кириллицы *(*А... Я, а...я*),* арабские цифры 0–9, знаки арифметических операций (+ /\ ^ * =) и знаки пунктуации (, ; . : ' _ () " ?), символы определения типа данных (& ! # $).

Константы – это величины, значения которых постоянны и не изменяются при выполнении программы. В Бейсике используют целые и вещественные числовые константы.

Числовые константы включают: цифры от 0 до 9; знак «минус» или «плюс» (последний принимается по умолчанию и, как правило, не указывается); десятичную точку, отделяющую целую и дробную части числа; буквы *E* или *D* между цифрами. Целочисленные константы – это целые числа (без дробной части). Числовые вещественные константы с фиксированной точкой содержат дробную и целую части, например 15.12, 2014.35, –123.45678.

Числовые вещественные константы с плавающей точкой записывают в экспоненциальной (нормализованной) форме: 0.25E02, –2.5E-03, что соответствует математической записи $0{,}25 \cdot 10^{2}$ и $-2{,}5 \cdot 10^{-3}$. Здесь знак константы определяется знаком мантиссы, а ее значение равно значению мантиссы, умноженному на 10 в степени, показатель которой записан после буквы *E*.

Переменные – это именованные величины, которые могут изменять свое значение в процессе выполнения программы.

Имя переменной (идентификатор) определяет ячейки области памяти компьютера, в которых хранится ее значение, и состоит из латинских букв и цифр, причем начинается обязательно с буквы.

2.2 Выражения и стандартные функции языка ТВ

Арифметические выражения в ТВ соответствуют общепринятым алгебраическим выражениям, в них могут входить числа, переменные, функции (стандартные или определяемые в программе), соединенные знаками арифметических операций.

К арифметическим операциям относятся: сложение (+), вычитание (–), умножение (*), деление (/), возведение в степень (^), целочисленное деление с отбрасыванием дробной части результата (\), вычисление остатка от деления целых чисел (**MOD**). Например, результат операции 11\2 есть 5, а результат операции 11 **MOD** 2 есть 1.

Арифметические выражения состоят из отдельных *операндов,* соединенных знаками арифметических операций. В качестве операндов могут выступать числовые константы и переменные, а также встроенные математические функции. При записи арифметических выражений необходимо учитывать приоритет выполнения операций, а именно:

1) вычисления в скобках;
2) вычисление встроенных функций;
3) возведение в степень;
4) умножение и деление;
5) целочисленное деление;
6) вычисление целого остатка отделения;
7) сложение и вычитание.

Если в выражении последовательно записать несколько операций с одинаковым приоритетом, то они выполняются слева направо в порядке их написания, за исключением встроенных функций и возведения в степень, которые выполняются справа налево.

Приведем примеры записи математической и на языке Бейсик выражений, содержащих различные арифметические операции (таблица 2).

Таблица 2

Математическая запись	Запись в Бейсике
$\frac{a}{b \cdot c}$	$a/(b*c)$ или $a/b/c$
$x^{1/3} + \sqrt{Lg\lfloor x \rfloor}$	$x\text{^}(1/3) + SQR(LOG(ABS(x)))$
$e^{-KX}\cos(ax)$	$EXP(-k*x)*COS(a*x)$
$\sin(x^2) + \cos^2(x)$	SIN(x^2)+COS(x)^2

Таблица 3

Функция	Математическая запись	Обращение в программе	Примечание
Синус	sin x	SIN(X)	X в радианах
Косинус	cos x	COS(X)	X в радианах
Тангенс	tg x	TAN(X)	X в радианах
Арктангенс	arctg x	ATN(X)	результат в радианах
Логарифм натуральный	ln x	LOG(X)	$X > 0$
Логарифм десятичный	lgx	LOGIO(X)	$X > 0$
Абсолютное значение	|x|	ABS(X)	
Корень квадратный	$\sqrt{}$	SQR(X)	
Экспонента	e^x	EXP(X)	
Наибольшее целое, не превосходящее x	[x]	INT(X)	

Логические выражения в основном используют для сравнения значений величин. Результатом логического выражения является "истина" (если отношение верно) или «ложь» (в противном случае). Простые логические выражения состоят из операции *отношения:* меньше (<), больше (>), равно (=), меньше или равно (<=), больше или равно (>=), не равно (< > или > <).

При написании сложных логических выражений участвуют операции, выполняемые над логическими величинам «истина» и «ложь»: **NOT** (логическое отрицание «не»), **AND** (логическое «и») и **OR** (логическое «или»). В сложном логическом выражении сначала выполняются логические отноше-

ния, а затем – логические операции согласно их приоритетной последовательности: **NOT**, **AND** и **OR**.

Примеры записи логических выражений в таблице 4.

Таблица 4

Математическая запись	Запись в Бейсике a > b
$a > b$	$a > b$
$b^2 - 4ac \geq 0$	$b\text{^}2 - 4 * a * c >= 0$
$d \neq 0$	$d <> 0$
$0 < f < 1$	$(f > 0) AND (f < 1)$

Операторы языка Бейсик представляют собой набор зарезервированных слов, которые задают команды компьютеру на выполнение действий для реализации алгоритма решения задачи.

Программа на языке Бейсик состоит из отдельных строк и при отсутствии явных команд, изменяющих последовательность ее действия, выполняется построчно по мере следования операторов. В каждой строке программы допускается присутствие нескольких операторов, отделенных друг от друга двоеточием «:». Любая строка может содержать *комментарий* – любой текст, начинающийся с апострофа «'» (символа, являющегося признаком комментария) или оператора **REM**.

Программу рекомендуется завершать оператором **END**, однако, он не является обязательным.

2.3 Основные операторы алгоритмического языка ТВ

2.3.1 Оператор присваивания

Оператор присваивания служит для присваивания переменной значения выражения, которым может быть арифметическое выражение, константа или имя другой переменной. Оператор присваивания имеет вид: **переменная = выражение**, например,

```
A=B^2+SIN(X);
C=D;
I=I+1; Y=12.
```

При употреблении оператора присваивания необходимо следить, чтобы к его моменту выполнения переменные, входящие в выражение правой части, были определены (имели числовые значения). Выражение в правой части вычисляется и присваивается переменной в левой части (относительно символа «=»).

2.3.2 Операторы ввода данных

Ввод исходных данных в программе осуществляется: оператором присваивания, оператором **READ** чтения данных блока, созданного оператором **DATA**; с клавиатуры во время выполнения программы при использовании оператора ввода **INPUT.**

Операторами присваивания ввод исходных данных осуществляется следующим образом:

A=5.2:B=12:X=–0.075.

Операторы **READ** и **DATA** в программе всегда присутствуют одновременно. При этом оператором **DATA** создается блок данных из констант, который затем читается оператором **READ**.

READ *список переменных*
DATA *список констант*

Здесь **READ** и **DATA** – служебные слова, *список переменных* – имя одной или нескольких разделённых запятыми переменных, *список констант* – значение одной или нескольких разделенных запятыми констант.

При выполнении оператора **READ** данные области **DATA** по очереди пересылаются в переменные, указанные в списке ввода оператора **READ**. Операторы **DATA** могут располагаться в любом месте программы, но в этих же строках не допускается размещение других операторов.

Ввод исходных данных, приведенных выше, с помощью оператора **READ, DATA** может быть осуществленных следующим образом:

READ A, B, X
DATA 5.2, 12,-0.075

При использовании оператора ввода **INPUT** исходные данные вводятся с клавиатуры непосредственно во время выполнения программы. Общий вид оператора:

INPUT *список вводимых переменных*

В операторе **INPUT** при необходимости можно привести текст подсказки для разъяснения последовательности ввода данных, тогда оператор имеет вид:

INPUT «*текст подсказки***»** *список переменных*

Список переменных – имена одной или нескольких переменных, отделенных друг от друга запятыми, значения которых необходимо ввести с клавиатуры.

Оператор **INPUT** приостанавливает выполнение программы в ожидании ввода данных и на экран выдает знак вопроса «?» или текста подсказки (если она приведена в операторе). После приостановки выполнения программы пользователь набирает на клавиатуре значения переменных в том порядке, в котором они указаны в *списке переменных,* отделяя их друг от друга запятыми или пробелами. После набора всех значений и нажатия клавиши «Enter» введенные данные присваиваются переменным, и выполнение программы продолжается. Если клавиша «Enter» ошибочно нажата до завершения набора данных, то оставшимся переменным списка присваивается значение нуля.

Пример пользования оператора **INPUT** при вводе данных с клавиатуры:

INPUT «Задайте три стороны треугольника**», A, B, C**

Во время выполнения этого оператора на экране появится текст и мигающий курсор в позиции, обозначенной ниже знаком «_».

*Задайте три стороны треугольника_*

В ответ пользователь вводит цифры, например 3 4 5, разделяя их пробелами или запятыми, и нажимает клавишу «Enter», в результате чего переменные получат значения: A=3, B=4, C=5.

Использование оператора **READ, DATA** целесообразно во время отладки программы, когда она многократно выполняется с одними и теми же исходными данными. Когда же программа многократно выполняется с различными исходными данными, целесообразно использование оператор **INPUT,** так как иначе перед каждым выполнением программы требовалось бы изменить оператор **DATA** в тексте программы.

2.3.3 Оператор вывода

Используется для вывода на экран дисплея или на принтер данных и результатов вычислений. Общий вид оператора:

PRINT *список*

Здесь **PRINT** – ключевое слово; *список* –- перечень констант, переменных или выражений, значения которых необходимо вывести на экран. Все элементы списка отделяются друг от друга запятой или точкой с запятой. Если они отделены *точкой с запятой,* то их очередные значения выводятся в одной строке с пробелом после каждого выведенного ранее значения. Если элементы списка разделены *запятой,* то их значения выводятся в отдельные зоны строки, каждая из которых занимает 14 позиций.

Например, в результате выполнения оператора вывода

PRINT *"Результат вычисления* = "; Y

на экране будет выведен следующий текст:

Результат вычисления = 24,076

Если требуется задать точный формат вывода отдельных элементов, то используется оператор вывода **PRINT USING**, который имеет вид: **PRINT USING** *«формат», список,* где формат задает образец распечатки элементов списка вывода.

При выводе числовых значений предусмотрен символ # для каждой выводимой цифры, а также знака числа. Все остальные символы, включая пробелы, обозначают сами себя и выводятся без изменений. Для вывода числа, содержащего дробную часть, в формате указывается столько символов #,

сколько всего цифр требуется вывести, и точка для отделения цифр целой части от дробной. Таким образом, использование формата позволяет отсечь при печати ненужные цифры, результат при этом округляется. Например, при выполнении оператора

PRINT USING "##. ## #. # ", *– 7.528, 2.634*

на экране получим

-7.53 2.6

Если в формате указан только один образ, то он будет использоваться многократно для каждого элемента списка вывода, при этом каждый элемент будет выводиться в новую строку. Например, при использовании

PRINT USING "##. ## ", *24.093,1.518, -1.11*

на экране получим

24.09

1.52

-1.11

Таким образом, оператор **PRINT USING** может обеспечить вывод данных в любом желаемом формате.

2.4. Программирование линейных алгоритмов

Рассмотрим реализацию простейшего линейного алгоритма на языке ТВ, ***Пример*** *2.* **/.** Вычислить значения $Y = a^2 + cos^2 b - |x - a|$ и $Z = y^3 + sin\, y^2$ для заданных значений переменных *a, b, x.*

```
PRINT “Введи A, B, X”
INPUT A, B, X
Y=A ^ 2 + COS(B)^2 –-ABS(X-A)
Z=Y^3+SIN(Y^2)
PRINT ”Y=”; Y ; “Z=”; Z
END
```

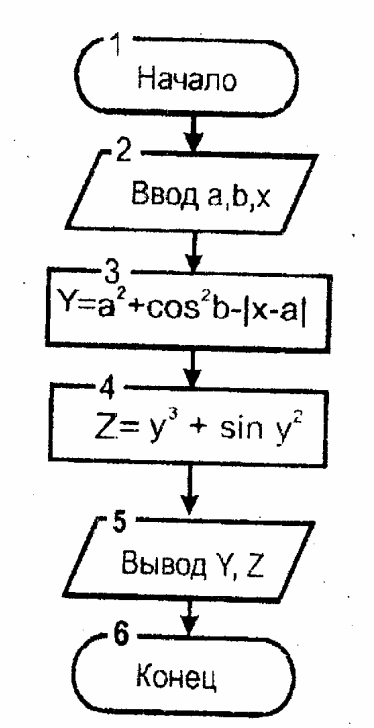

Рисунок 2 Схема алгоритма линейной структуры (пример 2.1)

При запуске программы на выполнение на экране появится текст «введи a, b,», после которого вводятся числовые значения исходных данных, разделенные пробелами или запятыми. После нажатия клавиши «Enter» на экране появится результат <y= <число> z = <число>

2.5 Основные операторы алгоритмического языка ТВ

Общий вид оператора:

GOTO n,

где *n* – номер строки (метка).

Оператор безусловного перехода **GOTO n** изменяет порядок выполнения программы, указывая номер строки, на которую без всякого условия необходимо передать управление для продолжения выполнения программы. Например, после выполнения оператора **GOTO 100** сразу начнет выполняться оператор, записанный в строке с номером 100. Следует отметить, что у большинства версий языка Бейсик строки в программе могут не нумероваться. Тем не менее, номерами (целыми числами) обязательно следует обозна-

чить те из них, на которые сделаны ссылки в операторах условного и безусловного переходов. Номер строки в таком случае называют меткой. В схемах алгоритмов действие данного оператора обозначается стрелкой.

2.6 Программирование алгоритмов разветвляющейся структуры. Оператор условного перехода

Оператор условного перехода **IF** служит для изменения порядка выполнения операторов в зависимости от какого-то условия. В качестве условия может быть записано выражение или логическое отношение. Оператор условного перехода **IF** может быть использован для организации разветвлений и циклов. Общий вид оператора **IF**:

IF < *условие* > **THEN** < *оператор 1* > **ELSE** < *оператор 2* >.

Действие условного оператора **IF** реализуется схемой следующего вида (рисунок 3).

Во время исполнения оператора **IF** вначале анализируется условие, записанное после ключевого слова **IF**, и если оно выполняется (истинно), то управление передается оператору (группе операторов, отделенных друг от друга двоеточием), приведенному после служебного слова **THEN**. Если условие не соблюдено (ложно), то выполняется оператор (операторы), приведенный после служебного слова **ELSE**. Затем выполняются операторы, следующие за оператором **IF**.

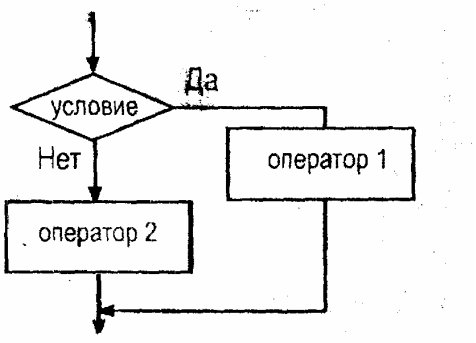

Рисунок 3 Схема условного оператора **IF**

Пример 2.2. При заданных значениях *x* и *y* вычислить значение *Z* по одной из формул:

$$z=\begin{cases} x^2+\sin y, \text{ если } x < y \\ y^3 - \cos, \text{ если } x \geq y \end{cases}$$

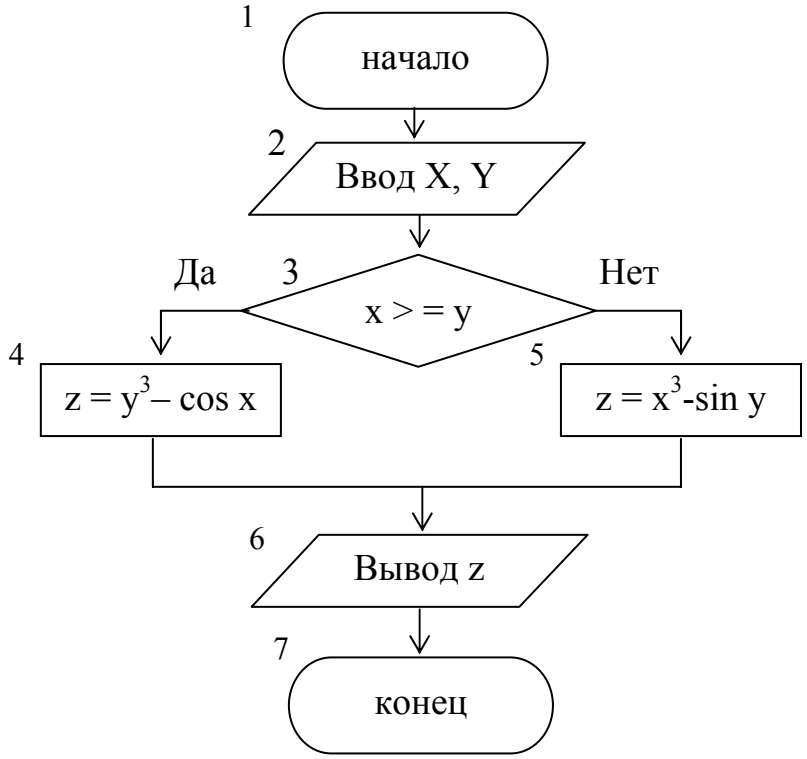

Рисунок 4 Схема алгоритма решения примера 2.2

```
PRINT“Введите X,Y”
INPUT X,Y
IF X>=Y THEN
      Z=X^2+SIN(Y)
      ELSE
      Z=Y^3-COS(X)
PRINT “Z=”;Z
END
```

Пример 2.3 Для заданных значений переменных *a, b* вычислить *Z*:

$$Z=\begin{cases} a^2+b, & \text{если} \quad a \leq -15 \\ a+b/2, & \text{если} \quad -15 < a < 0 \\ \sin a^2+b, & \textit{если} \quad 0 \leq a \leq 9 \\ \cos b^2+a^3, & \textit{если} \quad a > 9 \end{cases}$$

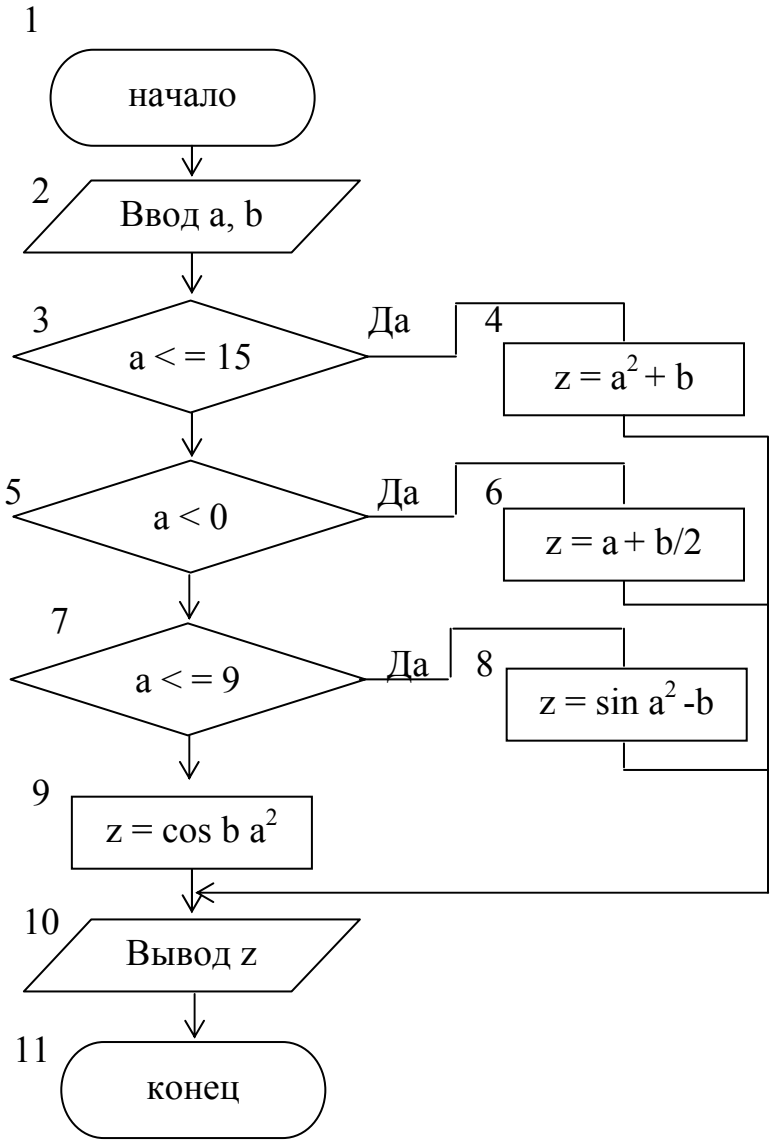

Рисунок 5 Схема алгоритма решения примера 2.3

```
PRINT“Введите A, B”
INPUT A,B
IF A<=15 THEN
      Z=X^2+B:GOTO 20
IF A<0 THEN
      Z=A+B/2:GOTO 20
IF A <=9 THEN
      Z=SIN(A^2)+B
      ELSE
      Z=COS(B^2)+A^3
20 PRINT “Z=”;Z
      END
```

У оператора **IF** ветвь **ELSE** может отсутствовать, тогда оператор записывается в виде:

IF *условие* **THEN** *оператор 1*

и реализуется схемой, представленной на рисунок 6.

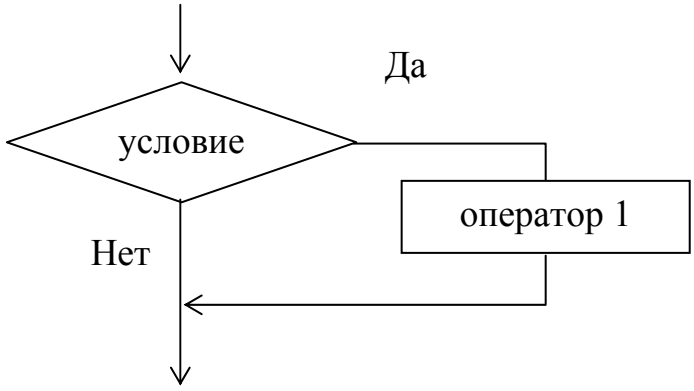

Рисунок 6 Схема условного оператора IF при отсутствии ветви ELSE

Если условие удовлетворяется (ветвь «Да»), то выполняется оператор 1, записанный за ключевым словом THEN и управление передается оператору, непосредственно следующему за IF. Если условие не выполняется (ветвь «Нет»), управление сразу же передается следующему оператору.

Рассмотрим организацию циклических вычислений с помощью оператора условного перехода **IF.**

Пример 2.4 Вычислить и вывести на печать все значения аргумента *x* и функции $y = x^2 + x \cdot \sin x$ при изменении *x* от 1 до 20 с шагом 0,25.

```
X=1
5 Y=X^2+X*SIN(X)
PRINT X;Y
X=X+0.25
IF X <=20 THEN 5
END
```

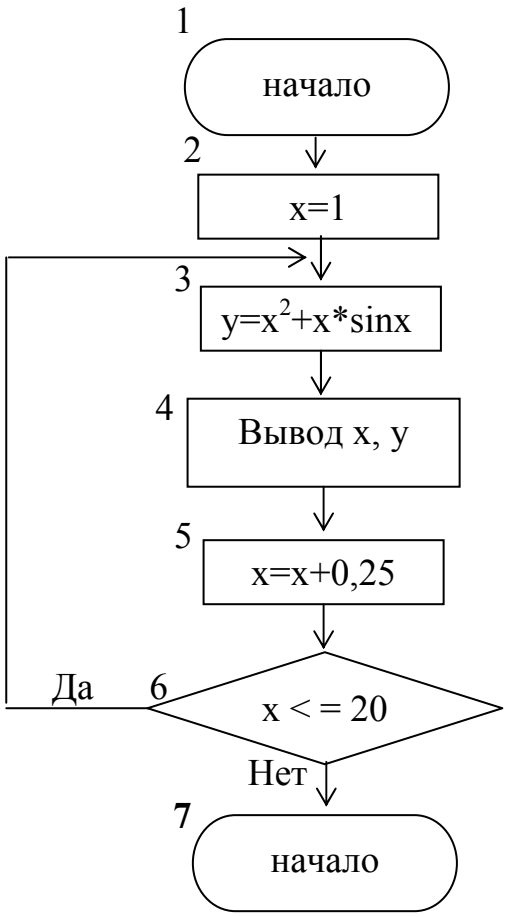

Рисунок 7 Схема алгоритма решения примера 2.4

2.7 Организация циклических вычислений. Операторы цикла FOR … NEXT

Циклические вычисления могут иметь известное или неизвестное из исходных данных число повторений вычислений. Циклы, в которых число повторений вычислений известно из исходных данных, называются арифметическими, в противном случае итерационными.

В Бейсике для организации арифметических циклических вычислений используется специальные операторы цикла FOR… NEXT. Общий вид операторов цикла

FOR I=a TO b STEP c
операторы тела цикла
NEXT I

Здесь **I** – переменная цикла; *a* –- начальное значение переменной цикла; *b*–- конечное значение переменной цикла; *c* – величина приращения (шаг) переменной цикла при каждом повторении цикла. Если шаг равен единице, запись **STEP** *c* – можно опустить. Верхней и нижней границами цикла служат операторы начала **FOR** и конца цикла **NEXT**. Операторы тела цикла выполняются многократно до тех пор, пока значение переменной цикла не превысит значение, заданное переменной *b.*

Величина шага переменной цикла может быть целым числом, числом с дробной частью, отрицательным или положительным, если шаг отрицательный, то начальное значение переменной должно быть больше конечного значения. Схема, реализующая операторы цикла, имеет вид, представленный на рисунок 8.

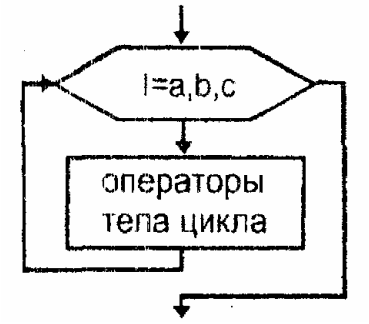

Рисунок 8 Схема операторов цикла FOR … NEXT

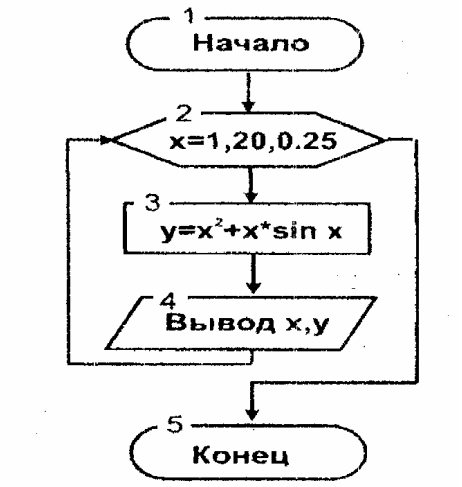

Рисунок 9 Схема алгоритма решения примера 2.4 с помощью операторов цикла

Рассмотрим решение примера 2.4 с помощью оператора цикла.

```
FOR X=1 TO 20 STEP 0.25
Y=X^2+X*SIN(X)
PRINT “X=”;X; “Y=”;Y
NEXT X
END
```

2.8 Программирование алгоритмов циклической структуры с использованием массивов данных

Массив – это совокупность однотипных данных, имеющих одно общее имя и индивидуальные порядковые номера (индексы), изменяющиеся от 1 с шагом 1. Имя массива образуется так же, как имена переменных (буква или буква и цифра). Положение элемента в массиве определяется индексами, записываемыми в круглых скобках после имени массива. Под массивы, так же как и под переменные, должен быть отведен определенный объем памяти. Для этого необходимо сообщить, какие массивы будут использоваться в программе и размер каждого массива. Эта информация задается в операторе описания массивов **DIM**. В операторе **DIM** указывается имя массива (или через запятую имена) и в круглых скобках задаются верхние границы изменения индексов, которые должны быть целыми положительными числами. В Бейсике могут использоваться массивы как числовые, так и символьные. Например, оператор

DIM A(20), B(5,10), C$(14)

описывает два числовых массива: одномерный **A** из 20 элементов, двумерный **B** из 5 строк и 10 столбцов и символьный массив **C$** из 14 элементов.

Ввод и вывод массивов осуществляется с использованием операторов цикла **FOR... NEXT**, переменная цикла при этом отслеживает изменения индексов каждого элемента. Ввод-вывод может быть организован как с помощью специально организуемого цикла, так и попутно с основным циклом вычислений. Например, при необходимости ввода и вывода одномерного массива **A** из 30 элементов ($\mathbf{A}_1$ $\mathbf{A}_2$,...,$\mathbf{A}_{30}$) фрагмент программы может иметь вид:

```
DIM A (30)
FOR I=1 TO 30
PRINT "Введите"; I; "-и элемент массива А";
INPUT A(I)
NEXT I
FOR J = l TO 30
PRINT J; "-й элемент массива А равен"; A (J)
NEXT J
```

В программе оператор **DIM A(30)** резервирует место для одномерного массива из тридцати элементов, которое затем заполняется числовыми значениями в порядке возрастания индексов. Использование первого оператора **PRINT** в программе позволяет организовать подсказку для пользователя, на устранение ошибок ввода. В подсказке по мере выполнения цикла переменная I в операторе **PRINT** указывает номер элемента массива, значение которого следует вводить. При выводе указывается значение каждого элемента и его место в массиве.

Пример 2.5 В заданном массивеA_1, A_2,… A_{30} найти среднее арифметическое отрицательных элементов массива, а затем вывести на печать элементы массива модуль, которых меньше модуля найденного значения, и найденное среднее арифметическое.

```
DIM A(30)
FOR 1=1 TO 30
INPUT A(I)
NEXT I
S=0: K=0
FOR 1=1 TO 30
IF A(I)<0 THEN S=S+A(I):K=K+1
NEXT I
SR=S/K
FOR 1=1 TO 30
IF ABS(A(I))<ABS(SR) THEN PRINT A(I)
NEXT 1
PRINT "Среднее арифметическое";SR
END
```

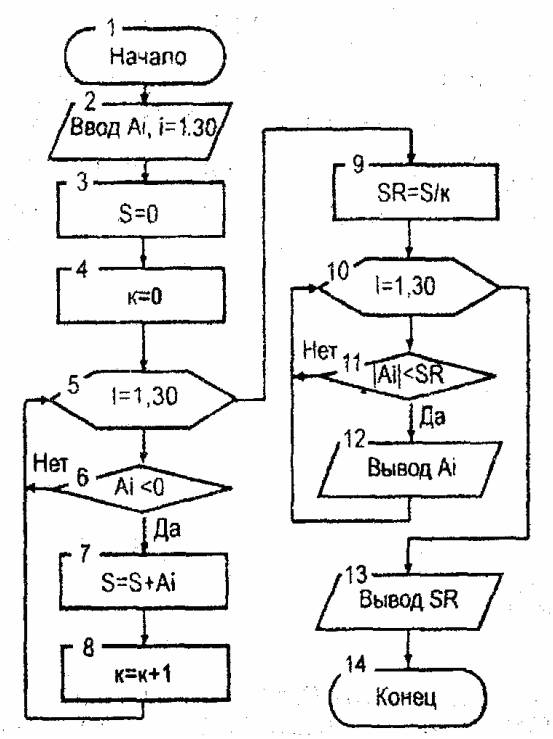

Рисунок 10 Схема алгоритма решения примера 2.5

Пример 2.6 Задано натуральное число n и массив X_1, X_2,… X_n. Превышает ли наибольший элемент массива число 25?

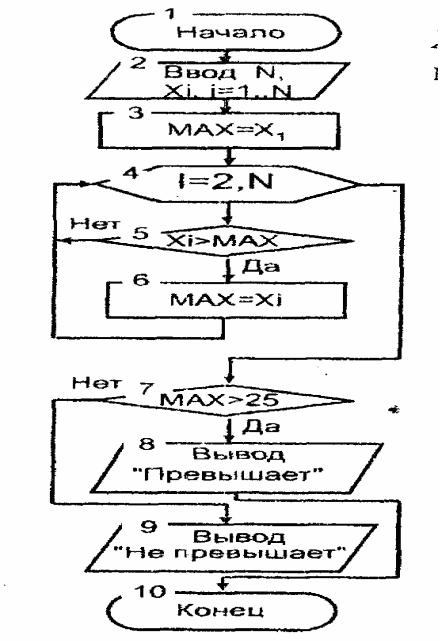

Рисунок 11 Схема алгоритма решения примера 2.6

```
PRINT ”Введи n”
INPUT N
DIM X (N)
FOR 1=1 TO N
INPUT X(I)
NEXT I
MAX=X(1)
FOR I=2 TO N
IF X(I)>MAX THEN MAX=X(I)
NEXT 1
IF MAX>25THEN
        PRINT "ПРЕВЫШАЕТ"
        ELSE
        PRINT "НЕ ПРЕВЫШАЕТ"
END
```

2.9 Алгоритмы и программы с использов анием итерационных циклов

Итерационными называют циклы, у которых количество вычислений в цикле предварительно неизвестно, а вычислительный процесс продолжается до выполнения определённого условия.

Пример 2.7 В ёмкости находится V $м^3$ жидкости. После работы в течение 1 часа откачивающего насоса производительностью P1 $м^3$ /час дополнительно подключился подающий насос производительностью P2 $м^2$ / час (причём P1 > P2). Определить, через сколько полных часов от начала работы 1-го насоса ёмкость опустеет, если кроме того в конце каждого часа порционно расходовалось P % от содержимого ёмкости.

Математическая формулировка задачи:

V = V− P1 − содержание ёмкости после 1-го часа работы откачивающего насоса.

$V = V - V \cdot P/100$ − содержимое ёмкости в результате ежечасного порционного расхода. V = V− P1 + P2 − содержимое ёмкости после каждого часа работы насосов. Тогда обозначим время через T, схема алгоритма решения задачи будет следующей (рисунок 12):

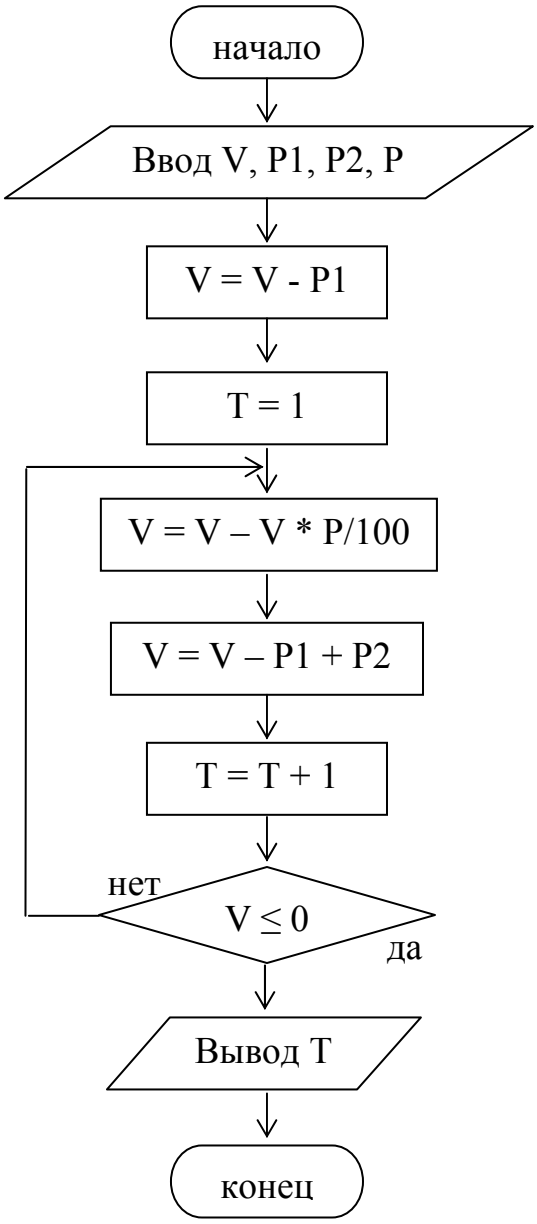

Рисунок 12 Схема алгоритма решения к примеру 2.7

Проверка на отсутствие жидкости в ёмкости осуществляется сравнением $V \leq 0$, так как при сравнении V равно нулю может произойти зацикливание вычислительного процесса.

Для программирования итерационных вычислений использование операторов цикла FOR … NEXT невозможно. В данном случае используются операторы присваивания и условного перехода IF.

```
      PRINT ‘’ Введи V, P1, P2,  P’’
      INPUT V, P1, P2, P
      V=V-P1
      T=1
2     V-V-V*P/1ØØ
      V=V-P1+P2
      T=T+1
      IF V<=0 THEN GOTO 5 ELSE GOTO 2
5     PRINT ‘’Через’’; T; ‘’ часов ёмкость опустеет’’
      END
```

Методические указания по выполнению задания 1

Задание 1 включает разработку схемы алгоритма решения задачи и составление программы на алгоритмическом языке Бейсик.

Студент должен выполнить индивидуальное задание 1 по своему варианту, номер которого совпадает с индивидуальным номером студента в учебном журнале группы.

Прежде чем приступить к решению задач, студент должен изучить соответствующие главы конспекта лекций и рекомендованной литературы и выполнить приведенные в них примеры и упражнения для полного освоения раздела.

При выполнении работы необходимо соблюдать следующие правила: условие каждой задачи переписать согласно варианту; схемы алгоритмов начертить в соответствии с рисунок 1 на решение задачи должно сопровождаться комментариями и пояснениями, тексты программ должны быть составлены на алгоритмическом языке Бейсик.

Варианты заданий

1. В электрической цепи с напряжением 220 вольт начальное сопротивление реостата составляет *Z* Ом. При каждом опыте это сопротивление уменьшают на *Q* Ом по сравнению с предыдущим значением. При каком значении со-

противления сила тока превысит величину A ампер и сколько опытов для этого потребуется произвести?

2. Тракторист в первый день работы вспахал S га поля. Затем каждый следующий день он увеличивал площадь вспашки на A га по сравнению с предыдущим днём. Через сколько дней он выполнит установленную норму вспашки Z га?

3. В течение 30 дней измеряли температуру воздуха и получили значения t_1, t_2, t_3, … t_{30}. Определить, превышает ли средняя температура месяца заданную величину Q и вывести на печать номер самого холодного дня.

4. Известно, что среди 20 сёл, в которых проживают t_1, t_1, … t_{20} жителей, есть сёла, удалённые от районного центра на расстояние, меньшее 15 км. Подсчитать количество таких сёл и общую численность жителей в них, если расстояния от каждого из сёл до райцентра составляют соответственно a_1, a_2… a_{20} (км).

5. Даны коэффициенты 40 точек (x_1, y_1), (x_2, y_2), … (x_{40}, y_{40}). Определить количество точек, лежащих во второй четверти, а также определить расстояние от наиболее удалённой точки среди всех до начала координат.

6. В бригаде 12 рабочих изготовили соответственно a_1, a_2, a_3,…..a_{12} изделий при плане Z изделий на каждого. Найти количество рабочих выполнивших план, среднюю выработку в бригаде и номер рабочего, выпустившего наименьшее количество изделий.

7. В бригаде из 12 рабочих которые выпустили соответственно x_1, x_2, …x_{12} изделий. Определить номер рабочего, выпустившего наибольшее количество изделий, среднюю выработку в бригаде и общее количество выпущенных изделий.

8. Две доярки обеспечили в течение месяца надои молока, характеризующиеся следующими показателями по дням:

Иванова – P_1, P_2, … P_{30} (литров),

Петрова – Q_1, Q_2,… Q_{30} (литров).

Составить алгоритм определения среднесуточных надоев молока каждой доярки в отдельности и обеими доярками вместе. Вывести на печать фамилию доярки, имеющий больший среднесуточный надой.

9. Тракторист за 10 дней вспахал соответственно a_1, a_2, a_3… a_{10} (га) пашни при плане P (га) в день. Определить среднюю выработку в день, общее количество вспаханной земли (в га) и номер дня, когда было вспахано наибольшее количество га.

10. В первый день месяца бригадой было выпущено p единиц продукции. Далее в течение месяца бригада каждый последующий день работы выпускает на L единиц продукции больше, чем предыдущий. Выполнит ли бригада месячную норму, которая составляет M единиц продукции? (Считать что в месяце 26 рабочих дней). Напечатать: «Бригада выполнила норму» или «Бригада не выполнила норму» в зависимости от полученного результата.

11. В бригаде 12 рабочих изготовили соответственно a_1, a_2, a_3,… a_{12} изделий при плане Z изделий на каждого. Вывести на печать номера рабочих, не выполнивших план, наибольшее количество изделий, выпущенное одним из рабочих бригады.

12. Ежедневные надои молока за месяц составили у первой доярки A_1, A_2… A_{30}, у второй B_1, B_2, B_{30} литров молока в день. Вывести на печать сообщение: у какой из доярок среднесуточный надой выше; количество молока, надоенное I, II доярками и общее количество молока.

13. Надои молока, полученные дояркой в течение месяца, характеризуются следующими показателями по дням: A_1, A_2,… A_{30} (литров). Вывести на печать дневные надои молока, полученные в те дни, когда выполнялась или перевыполнялась установленная ежедневная норма надоев молока, составляющая p (литров), а также определить количество молока, полученного за первую декаду месяца.

14. Заданы координаты 10 точек на плоскости: (x_1, y_1); (x_2, y_2); …(x_{10}, y_{10}). Определить, сколько точек попадает в круг радиуса R с центром в начале координат, и напечатать номера точек, не попавших в этот круг.

15. Заданы координаты 10 точек на плоскости: (x_1, y_1); (x_2, y_2); …(x_{10}, y_{10}) и круг радиуса R с центром в начале координат. Определить, сколько точек окажется внутри круга на окружности и вне круга.

16. По плану в 7 колхозов должны были доставить соответственно x_1, x_2, x_3,… x_7 новых комбайнов, а фактически поставили y_1, y_2, y_3, … y_7 комбайнов. Напечатать номера колхозов, в которые недопоставили комбайны и номер колхоза, получившего наибольшее количество комбайнов.

17. В массиве x_1, x_2,… x_{12} определить количество отрицательных элементов, порядковые номера элементов, равных нулю и среднее арифметическое положительных элементов.

18. Хозяйство имеет тракторный парк в количестве 35 тракторов. Среднедневная выработка каждого трактора составляет A_1, A_2,… A_{35} [га]. Определить количество тракторов и номера тракторов, среднедневная выработка которых больше заданной величины T (га) и общий объем работы, выполняемый этими тракторами за день.

19. Информация о температуре воздуха за месяц задана следующими величинами T_1, T_2,… T_{30}. Определить, сколько раз температура опускалась ниже нуля. Напечатать числа месяца, в которые это случалось.

20. По плану в 7 колхозов должны были доставить соответственно x_1, x_2, x_3, …x_7 новых комбайнов, а фактически поставили y_1, y_2, y_3,… y_7 комбайнов. Определить количество колхозов, в которые недопоставили комбайны и наименьшее количество комбайнов, доставленных в один из колхозов.

21. Тракторист за 10 дней вспахал соответственно a_1, a_2, a_3… a_{10} (га) пашни при плане P (га) в день. Определить среднюю выработку в день, общее количество вспаханной земли (га) и номера дней, когда не выполнялась плановое задание.

22. Задан массив a_1, a_2, a_3… a_{12} . Найти среднее арифметическое отрицательных элементов массива (предполагая, что такие элементы в массиве есть) и вывести на печать номера положительных элементов.

23. В цистерне находится Z литров топлива. В первый день израсходовали Q литров топлива, а затем в каждый последующий день стали расходовать на K литров больше, чем в предыдущий. Определить, через сколько дней цистерна опустеет.

24. В массиве x_1, x_2, x_3, … x_{10}, определить, каких элементов больше: отрицательных, положительных или равных нулю.

25. В банк положили Z рублей. В конце каждого месяца вклад увеличивается на Q **%** , а в конце каждого года вкладчик снимает B рублей (при этом вклад постоянно увеличивается). Через сколько месяцев вклад превысит $\boldsymbol{A}$ рублей.

26. В течение месяца бригада каждый последующий день работы выпускает на три изделия больше предыдущего. В первый день было выпущено 42 изделия, что составило 67 % дневной нормы. Выполнит ли бригада месячную норму? (Считать в месяце 24 рабочих дня). Напечатать одно из сообщений: «Бригада выполнит норму» или «Бригада не выполнит норму» в зависимости от полученного результата.

27. Имеется список 20 членов бригады колхозников с указанием их возраста A_1, A_2,… A_{20} . Определить средний возраст бригады и напечатать порядковые номера членов бригады, чей возраст превышает средний.

28. На сберкнижку положено $\boldsymbol{S}$ рублей. Определить сумму данного счета, а также суммарный его прирост по процентам по истечении n лет, если по окончании каждого года хранения сумма увеличилась на 2 %, а затем вкладчиком производилось изъятие со счета T рублей. ($T < (2/100)\cdot S$).

29. Массив k_1, k_2,… k_{20} содержит сведения о том, на сколько отличается объем выпущенной продукции каждым из 20 отстающих рабочих бригады от плановых заданий. Определить, кто из рабочих имеет наибольшее отставание, напечатать его порядковый номер, а также определить общее недовыполнение плана всеми рабочими.

30. Задан массив z_1, z_2, …z_9. Определить номер наименьшего элемента (считая, что он один) и среднее арифметическое отрицательных элементов массива.

ЗАДАНИЕ 2

Для нелинейного (алгебраического или трансцендентного) уравнения, приведённого в таблице 5, произвести отделение его корней и уточнить их с помощью разработанных алгоритмов и программ.

Аргументы тригонометрических функций принять в радианах, а уравнение выбрать в соответствии с последней цифрой шифра зачётной книжки и первой буквой фамилии.

Таблица 5

Варианты нелинейных уравнений к заданию 2.

Последняя цифра шифра	Первая буква фамилии		
	А, Г, Ж, Е, К, Н, Р, У, Ц, Ш, Ю	Б, Д, З, И, Л, О, С, Ф, Ч, Ы, Я	В, Е, М, П, Т, Щ, Х, Э
0	$x-\cos x=0$	$x-2+e^{x}=0$	$3x-\cos x=1$
1	$2-x=\ln x$	$2x-\cos x=0$	$4x-6\sin x=0$
2	$x^{3}-9x+3=0$	$x^{3}-10x+5=0$	$x^{3}-15x+8=0$
3	$x^{3}-8x+2=0$	$x^{3}-2{,}6x+1{,}5=0$	$x+\ln x=0{,}5$
4	$x^{3}-15x+10=0$	$x^{3}-20x-10=0$	$2x-\sin x-0{,}9=0$
5	$x^{3}-5x+2=0$	$x^{3}-2x-1=0$	$x^{3}+4x-3=0$
6	$e^{x}+x=0$	$x-\sin x=1$	$e^{x}+e^{-3x}=4$
7	$x-\sin x=0{,}25$	$\sin x-5x+1{,}5=0$	$3x-\cos x-1=0$
8	$x^{3}+10x+3=0$	$5x^{2}=2$	$2x+x^{2}-1{,}2=0$
9	$x^{3}+3x-1=0$	$2^{x}=4x+3$	$x^{3}+3x=2$

Методические указания к выполнению задания 2

При решении любое нелинейное уравнение, например $x^{3}-3x=1$, предварительно приводится к виду $f(x)=0$, т. е. $x^{3}-3x-1=0$, при этом функции $f(x)$ могут быть алгебраическими, у которых над аргументом *x* выполняются лишь арифметические операции, и трансцендентными, которые содержат, кроме того, показательные, логарифмические, тригонометрические функции аргумента *x*. Такие уравнения далеко не всегда можно решить точно, а для практических расчётов точное решение не всегда является обязательным.

Поэтому особое значение с развитием вычислительной техники приобрели способы приближённого вычисления корней и оценки их точности.

Данная процедура состоит из двух этапов: 1) отделение корней, т. е.. выделение таких отрезков [a, b], в каждом из которых находится только один корень уравнения или нахождение первоначальных приближений корней; 2) уточнение корней, т. е.. нахождение их на найденных отрезках с заданной степенью точности.

1. Простейшим способом отделения корней является графический. На графике функции $y = f(x)$ определяются отрезки [a, b], в пределах которых лежат точки пересечения функции с осью 0x, являющимися приближенными значениями корней. Например, при решении уравнения $x^3 - 3x - 1 = 0$ сначала вычисляем значения функции $y = x^3 - 3x - 1$, по заданным значениям аргумента x (таблица 6) и затем по полученным данным строим график (рисунок 12).

Таблица 6

Результаты табулирования функции $y = x^3 - 3x - 1$

x	–3	–2	–1	0	1	2	3	4
y	–19	–3	1	–1	–3	1	17	51

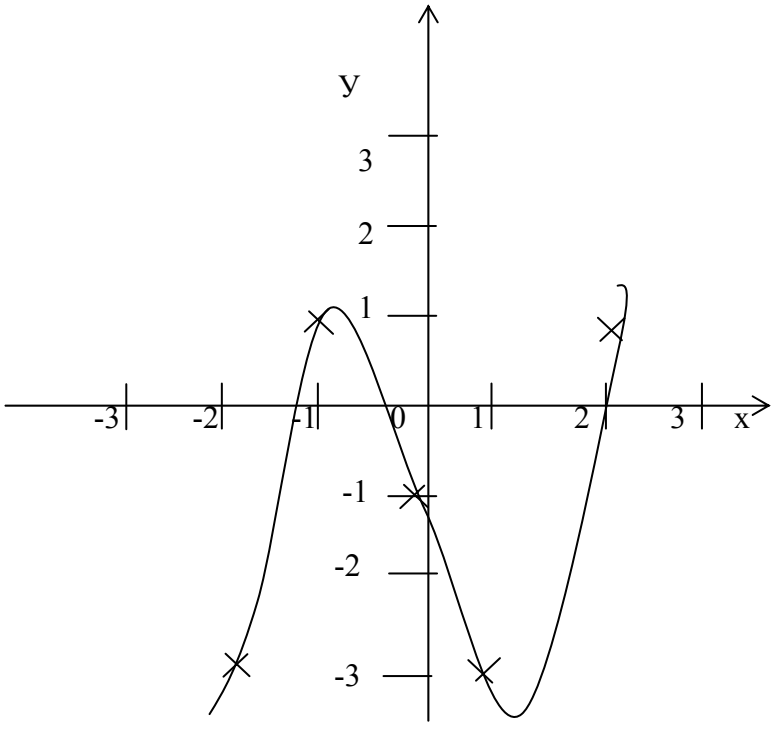

Рисунок 12 График функции $y = x^3 - 3x - 1$

Как видно из графика, уравнение имеет 3 корня на отрезках [–2, –1], [–1,0] и [1,2].

Отделить корни можно также программным способом. Для составления алгоритма решения данной задачи рассмотрим поведение функции $y = f(x)$ на интервале от $x_{нач}$ до $x_{кон}$ с шагом h (рисунок 13)

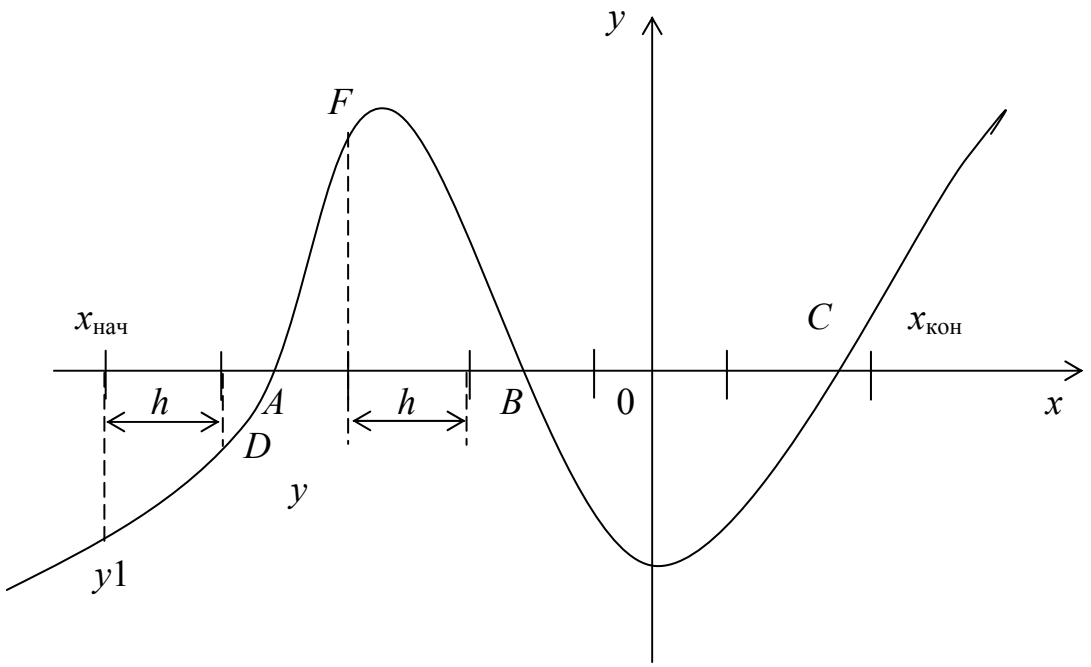

Рисунок 13 График функции $y = f(x)$ при наличии корней

Как видно из рисунка, корни уравнения находятся в точках A, B и C, где $f(x) = 0$. Данные точки можно обнаружить, рассчитывая значения функции $y = f(x)$ от $x_{нач}$ до $x_{кон}$ с шагом h и наблюдая за изменением её знака. Так, в точке A значение функции изменяется с «–» на «+», а в точке B с «+» на «–» и т. д. Обнаружить изменение знака функции можно путём перемножения двух её значений, вычисленных от аргументов x, находящихся в соседних точках на расстоянии h одно от другого. Для точки A, например, это будут значения функции D и F.

Отделение корней по рассмотренному методу реализует алгоритм, представленный на рисунке 14.

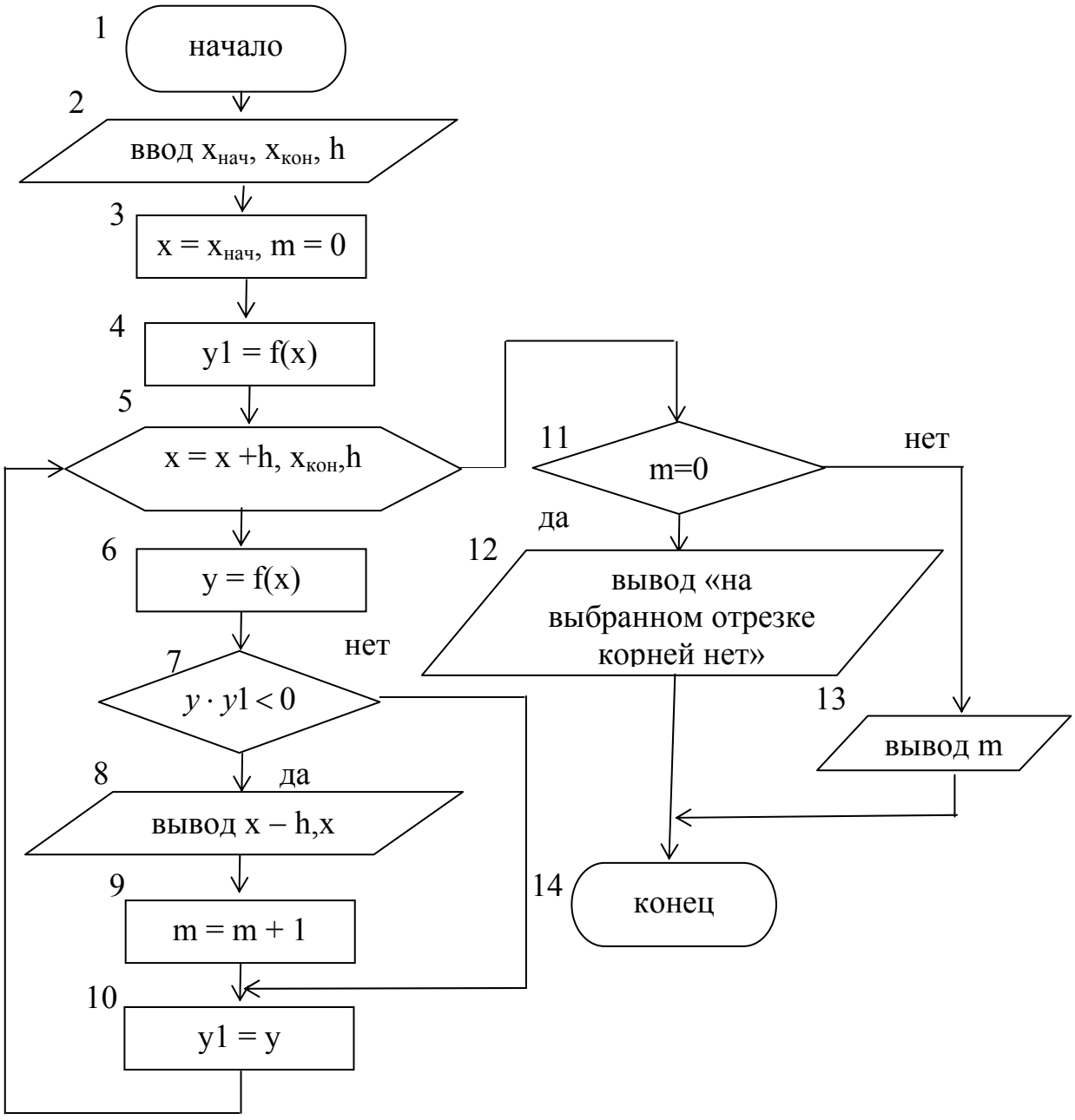

Рисунок 14 Схема алгоритм отделения корней функции f(x) на отрезке [$x_{нач}$, $x_{кон}$].

В блоке 2 вводятся начальное значение аргумента функции $x_{нач}$, конечное $x_{кон}$ и шаг его изменения h.

В блоке 3 принимается текущее значение x за $x_{нач}$ и количество корней $m = 0$, а в блоке 4 вычисляется первоначальное значение функции $y1$ при $x = x_{нач}$. Блок 5 организует цикл по изменению x от $x + h$ до $x_{кон}$ с шагом h. В блоке 6 вычисляется очередное значение функции в точке $x = x + h$ ($y = f(x)$). Затем в блоке 7 проверяется, не пересекла ли функция $y = f(x)$ ось OX. В случае

отсутствия пересечения $y \cdot y1 > 0$ полученное значение y в блоке 10 запоминается как y_1, а в блоке 5 значение x опять изменяется на величину шага h. Затем в блоке 6 снова вычисляется y и в блоке 7 проверяется функция на изменение знака. Так как в этом случае $y \cdot y1 < 0$ (точки D и F на рисунок 4), то в блоке 8 выводятся значения аргумента $x - h$ и x, в пределах которых находится корень уравнения, а в блоке 9 фиксируется увеличение количества найденных корней на единицу. Циклический процесс повторяется до принятого конечного значения аргумента $x_{кон}$. В результате будут найдены отрезки, на которых находятся корни нелинейного уравнения (около точек A, B, C на рисунок 13).

Программа на языке Basic, реализующая данный алгоритм для уравнения $x^3 - 3x = 1$, будет следующей:

```
PRINT ″Введите Xнач, Xкон, H″
INPUT XN, XK, H
X = XN: M = 0
Y1 = X*X*X– 3*X – 1
FOR X = X + H TO XK STEP H
Y = X*X*X – 3*X – 1
IF Y*Y1 < 0 THEN GOTO 2 ELSE GOTO 5
2 PRINT “A=”; X – H; “B=”; X
M = M + 1
5 Y1= Y
NEXT X
IF M = 0 THEN PRINT “корней нет” ELSE PRINT “корней=”;M
END
```

Уточнение корня, найденного на отрезке [a, b], осуществляется одним из следующих методов: деления отрезка пополам, хорд (секущих), касательных (Ньютона), итераций. Рассмотрим некоторые из них, например, метод деления отрезка пополам (рисунок 15). Интервал [a, b] делится пополам и в найденной точке ($c = (a + b) / 2$) вычисляется значение функции $y = f(c)$. Если $|y| \le e$, где e – заданная точность, то C является корнем уравнения, т.к. при полученном C функция $y = f(C)$ меньше точности e. В противном случае выбираем один из отрезков - или $[a, (a + b) / 2]$ (рисунок 15 *а*), или $[(a + b) / 2, b]$

(рисунок 15 *б*), на концах которого $f(x)$ имеет противоположные знаки. Выбранный интервал снова делим пополам ($c = (a + b) / 2$) и вычисляем значение функции $y = f(c)$. Процесс повторяется до тех пор, пока не будет получено значение $|y| \le e$.

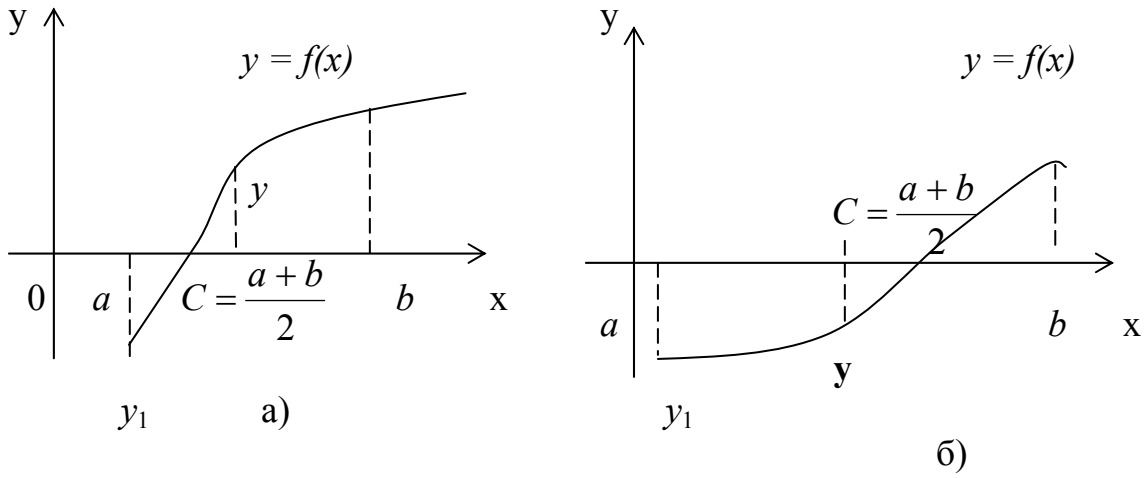

Рисунок 15 Уточнение корня методом деления отрезка пополам

Сказанное выше реализуется следующим алгоритмом (рисунок 16), где в блоке 2 вводятся полученные при отделении корней границы интервала [*a, b*] и точность вычисления корня e, а в блоке 3 вычисляется значения функции y_1 при $x = a$. Затем в блоке 4 вычисляется середина интервала [*a, b*], а в блоке 5 – значение функции в середине данного интервала при $c = (a + b) /2$. Если при проверке в блоке 6 оказывается $|y| \le e$, то c – корень уравнения, который выводится в блоке 10. Если же условие $|y| \le e$ не выполняется, то в блоке 7 определяется: какую половину отрезка [*a*, *b*] оставить для дальнейшего нахождения корня. Если $y \cdot y_1 < 0$, то левую присвоением b = c (блок 9), а если же нет, то правую присвоением a = c (блок 8) и затем в блоке 4 опять определяется середина нового суженного интервала и процесс повторяется до тех пор, пока значение y станет меньше заданной точности *e*.

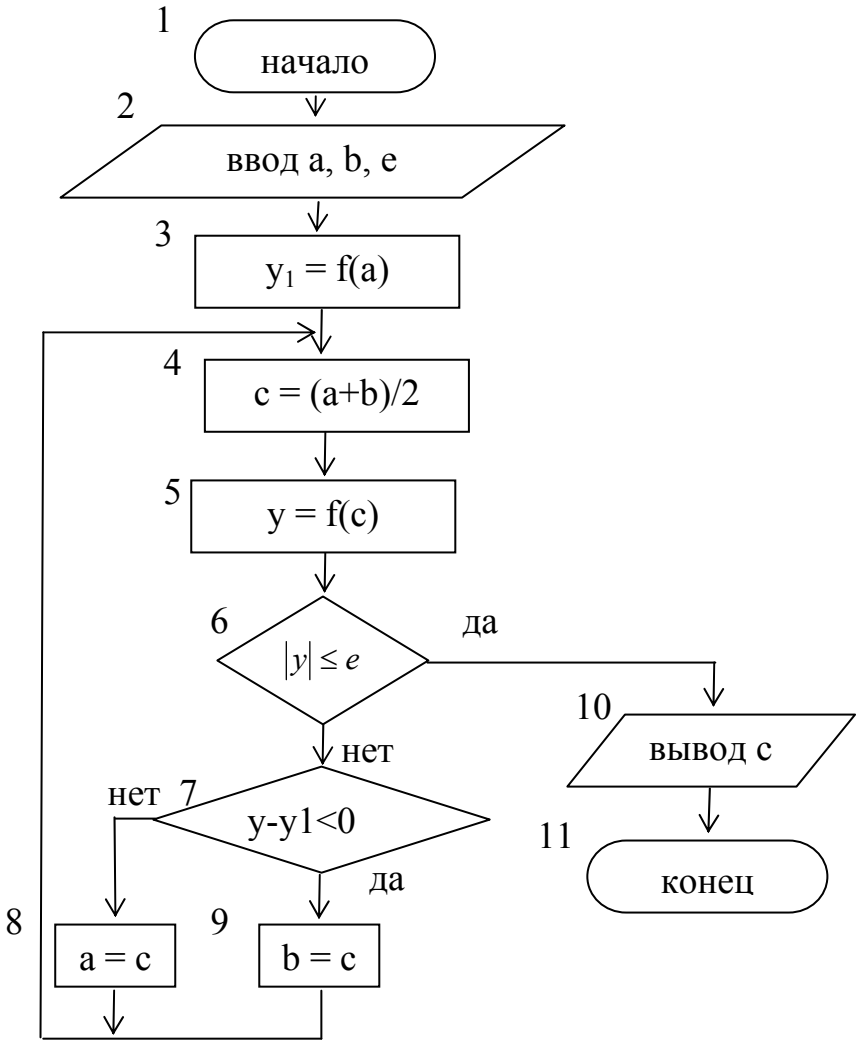

Рисунок 16 Схема алгоритма уточнения корня методом деления отрезка пополам

Программа на языке Basic, реализующая данный алгоритм для уравнения $x^3 - 3x = 1$, будет следующей:

```
     PRINT “Введите A, B, E “
     INPUT A, B, E
     Y1 = A*A*A – 3*A – 1
 4   C = (A + B)/2
     Y = C*C*C – 3*C – 1
     IF ABS(Y) < E GOTO 10
     IF Y*Y1< 0 THEN B = C ELSE A = C
     GOTO 4
10   PRINT “корень=”; C
     END
```

При уточнении корня методом итераций в уравнении неизвестное выражают через самого себя, т. е.. уравнение приводится к виду $x = f(x)$. Тогда рассмотренное выше уравнение $x^3 - 3x = 1$ преобразуем к виду $x = (x^3 - 1)/3$.

Выберем произвольную точку x внутри отрезка [*a*, *b*], на котором находится корень уравнения, и подставим это значение в правую часть преобразованного уравнения, получив соответственно $x_n = (x^3 - 1)/3$. Затем, приняв *x* равным полученному x_n ($x = x_n$), опять проведём вычисления нового x_n.

Этот процесс последовательного вычисления значений x_n по формуле будет продолжаться до тех пор, пока разность между вычисленным x_n и предыдущим x по модулю не станет меньше заданной точность *e* ($|x_n - x| \le e$). Рассмотренное выше нахождение корня реализуется следующим алгоритмом (рисунок 17).

Метод итераций применим только в том случае, если вычислительный процесс сходится (т. е. от итерации к итерации абсолютная разность $|x_n - x|$ будет уменьшаться. Для этого необходимо провести преобразования исходного уравнения к виду $x = f(x)$ так, чтобы выполнялось условие $|f'(x) < 1|$ для любого значения *x*, принадлежащего отрезку [*a*, *b*].

Для предотвращения зацикливания в случае расходящегося процесса в схему алгоритма блоком 2 вводится параметр *m*, обеспечивающий ограничение на максимальное число итераций. Количество итераций подсчитывается в блоке 5 и при превышении заданного числа m блок 7 прерывает процесс поиска корня. Уточнение корней методом касательных подробно рассмотрено в [4].

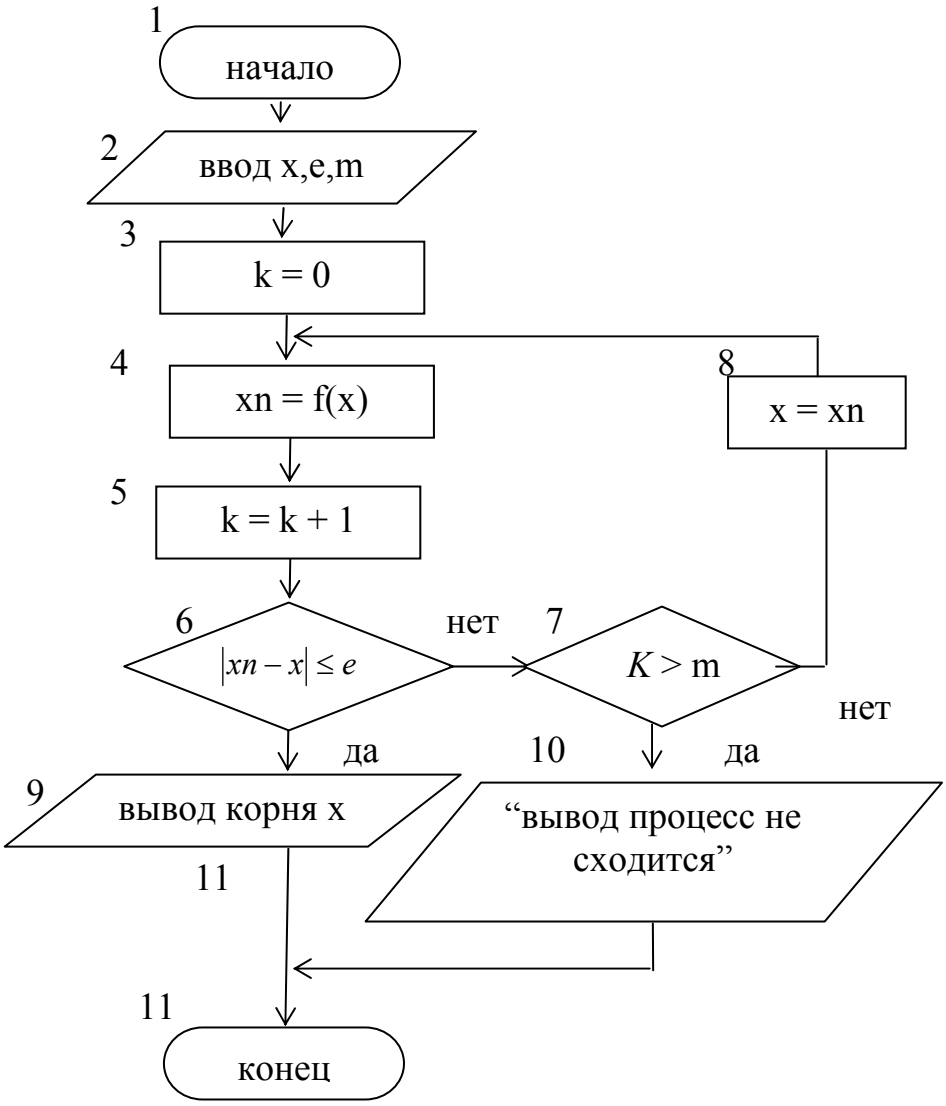

Рисунок 17 Схема алгоритма уточнения корня методом итераций

Вопросы для самоконтроля

1 Из каких этапов состоит процесс решения нелинейных уравнений?

2 Чем характерна область, где находится корень уравнения?

3 Как в алгоритме и программе определяется область нахождения корня?

4 В чем сущность нахождения корня с необходимой точностью в заданной области?

ЗАДАНИЕ 3

Составить алгоритм и программу вычисления определенного интеграла согласно таблице 7. Номер варианта принять в соответствии с последней цифрой шифра зачетной книжки и первой буквой фамилии. Метод вычисления и количество частей N выбрать по своему усмотрению.

Таблица 7

Варианты интегралов к заданию 3

Последняя цифра шифра	Первая буква фамилии					
	А, Г, Ж, В, К, Н, П, У, Ц, Ш, Ю, Б, Д, З, Л			О, С, Ф, Ч, Н, Я, В, Е, И, Р, Т, Щ, Э, Х, М		
	Подынтегральная функция	Пределы интегрирования		Подынтегральная функция	Пределы интегрирования	
		a	b		a	b
1	$2\sin x$	0	$\pi/2$	$5\sin x$	$\pi/2$	π
2	$4x^3$	1	2	$3x^2$	1	2
3	$3\cos x$	0	$\pi/2$	$5\cos x$	0	π
4	$2x^3$	0	2	$4\sin x$	0	π
5	$4\cos x$	$\pi/2$	π	$6x^2$	0	2
6	x^3	2	3	$2\cos x$	$\pi/2$	π
7	$3\sin x$	0	$\pi/2$	$2x^5$	1	4
8	$9x^2$	1	2	$6\sin x$	0	π
9	$6x$	3	4	$6x^2$	2	3

Вычисление определенных интегралов численными методами

Определенный интеграл от непрерывной функции $f(x) \geq 0$ в пределах от a до b представляет площадь криволинейной трапеции S, ограниченной кривой f(x), осью абсцисс и прямыми $x = a$, $x = b$ (рисунок 18). Из курса высшей математики известно, что

$$S = \int_a^b f(x)dx = F(x)\Big|_a^b = F(b) - F(a),$$

где $F(x)$ – первообразная для $f(x)$ на отрезке $[a, b]$, т. е.. $F'(x) = f(x)$ на отрезке [a, b]. Если $f(x) < 0$ на отрезке [a, b], то в формуле $S < 0$, но $|S|$ равно площади криволинейной трапеции, находящейся под осью абсцисс.

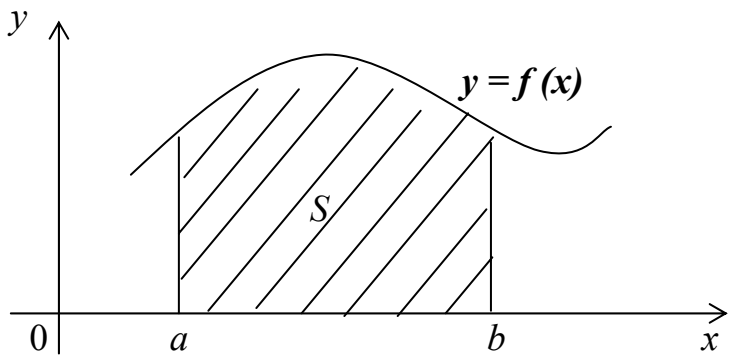

Рисунок 18 Определённый интеграл – площадь криволинейной трапеции

Однако на практике приведённой формулой часто нельзя воспользоваться по двум основным причинам:

1) вид функции $f(x)$ не допускает непосредственного интегрирования, т. е. первообразную нельзя выразить в элементарных функциях;

2) значения функции $f(x)$ заданы только на фиксированном конечном множестве точек x_i, т. е. функция задана в виде таблицы.

В этих случаях используются методы численного интегрирования. Они основаны на аппроксимации подынтегральной функции некоторыми более простыми выражениями, например, многочленами нулевой ($y = c$), первой ($y = cx + d$) или второй ($y = cx^2 + dx + k$) степени, а численные методы вычисления определенного интеграла, основанные на подобной аппроксимации, называются соответственно методами прямоугольников, трапеций и Симпсона (парабол).

Пусть требуется приближенно вычислить значение интеграла $S = \int_a^b f(x)dx$ В методе прямоугольников (рисунок 19) криволинейная трапеция разбивается на n частей, каждая из которых представляет собой прямо-

угольник, основание которого равно шагу интегрирования $h = \frac{b-a}{n}$, а длины сторон соответственно $Y_0 = f(x_0)$, $Y_1 = f(x_1)$, $Y_n = f(x_n)$, где $x_0 = a$, x_1, …, x_{n-1}, $x_n = b$ – точки деления отрезка $[a, b]$ на n равных частей.

Различают методы правых, левых и средних прямоугольников, в зависимости от месторасположения начальной точки x_0 при вычислении площади элементарного прямоугольника. При этом если за высоту каждого прямоугольника принимается левая ордината (y_0, y_1, y_2…), то вычисление интеграла будет производиться по методу левых прямоугольников; если правая ордината (y_1, y_2, y_3…), то по методу правых прямоугольников; если за высоту принимается середина интервала h, то будет применяться метод средних прямоугольников. Основанием всех прямоугольников будет являться величина шага интегрирования h.

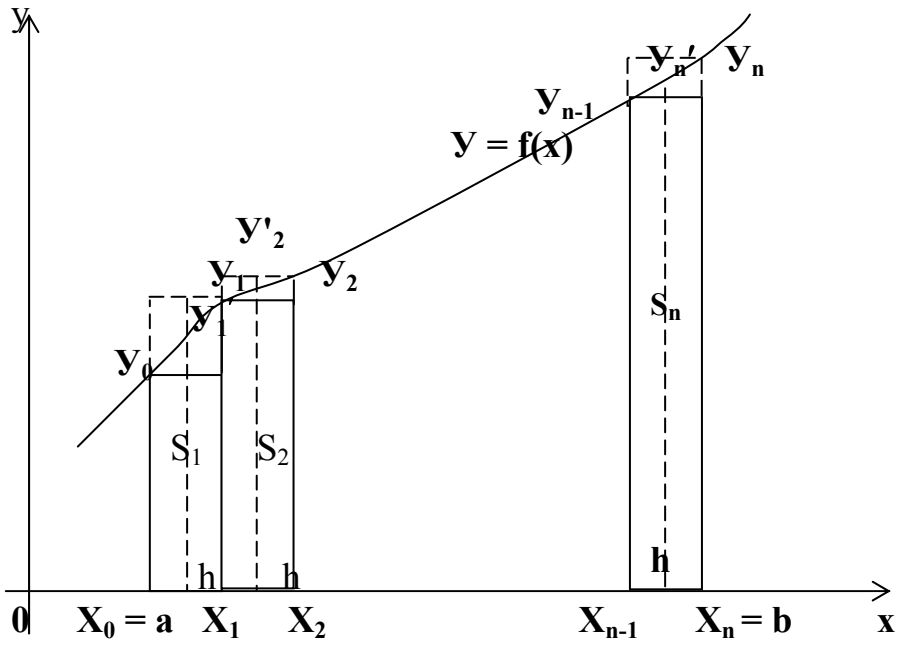

Рисунок 19 К выводу формул вычисления определённых интегралов методами прямоугольников

Тогда при методе левых прямоугольников

$$\int_a^b f(x)dx = S1 + S2 + ... + Sn = y_0 \cdot h + y_1 \cdot h + ... + y_{n-1} \cdot h,$$

при методе правых

$$\int_a^b f(x)dx = S1 + S2 + ... + Sn = y_1 \cdot h + y_2 \cdot h + ... + y_n \cdot h,$$

при методе средних

$$\int_a^b f(x)dx = S1 + S2 + ... + Sn = y_1^{/} \cdot h + y_2^{/} \cdot h + ... + y_n^{/} \cdot h.$$

Как видно из рисунка 19 первоначальное значение x при методе левых прямоугольников $x = a$, правых – $x = a + h$, средних $x = a + h/2$. Последующие значения x будут получаться через операцию присваивание $x = x + h$, а элементарные площади S_1, S_2,… Sn будут вычисляться по формуле $St = f(x) \cdot h$. Сумма этих площадей дает значение интеграла. Изложенное выше реализует алгоритм (рисунок 20), где St – значения элементарных площадей, а их сумма S – значение интеграла.

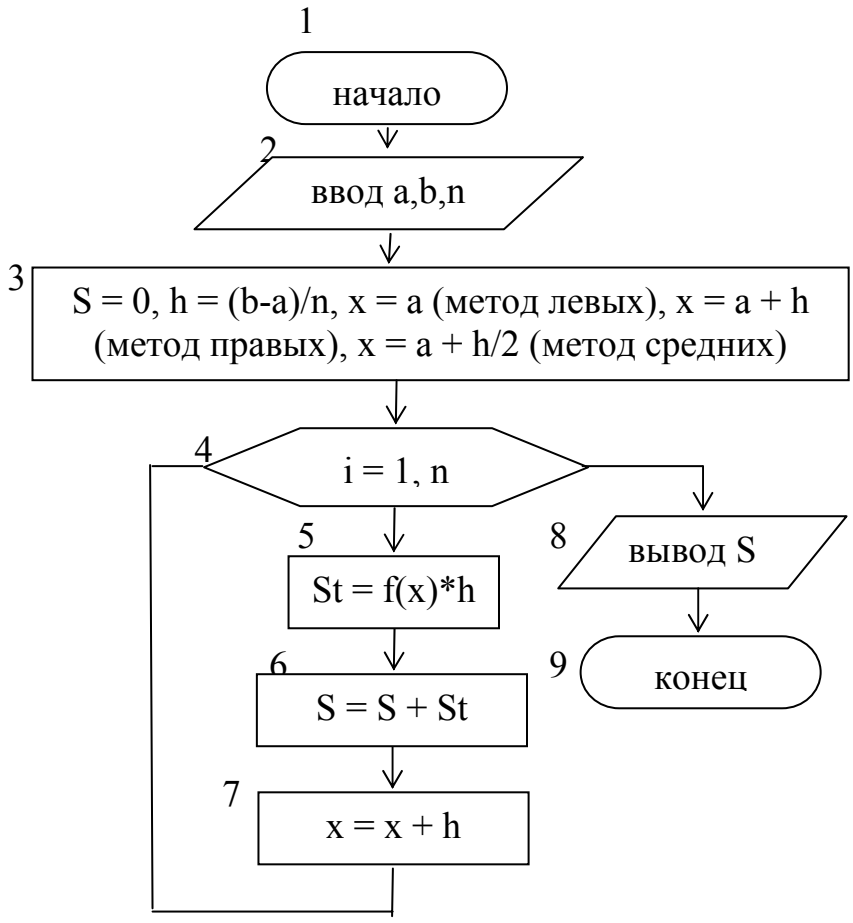

Рисунок 20 Схема алгоритма вычисления интеграла методом прямоугольников

Более точное значение интеграла получается при вычислении его методом трапеций, когда ординаты (y_0, y_1, y_2… y_n) подынтегральной функции со-

единяют отрезками прямых и искомую площадь заменяют суммой площадей трапеций, высотой которых является шаг h, а основаниями y_0 и y_1 для S_1, y_1 и y_2 для S_2 (рисунок 19).

Тогда

$$\int_a^b f(x)dx = S1 + S2 + \ldots + Sn = h\frac{y_0 + y_1}{2} + h\frac{y_1 + y_2}{2} + \ldots + h\frac{y_{n-1} + y_n}{2},$$

где $h = (b-a)/n$, а $y_0, y_1, y_2 \ldots y_n$ равны значениям функции $f(x)$ при соответствующих значениях аргумента x.

Поскольку $y_0 = f(a)$, $y_1 = f(a+h)$, $y_2 = f(a+2h)$ …, то схема алгоритма примет вид, приведенный на рисунке 21. В приведенном алгоритме блок 5 вычисляет значение элементарных площадей S_1, S_2,…S_n, в блоке 6 осуществляется их суммирование и блоком 7 изменяется x на величину шага h.

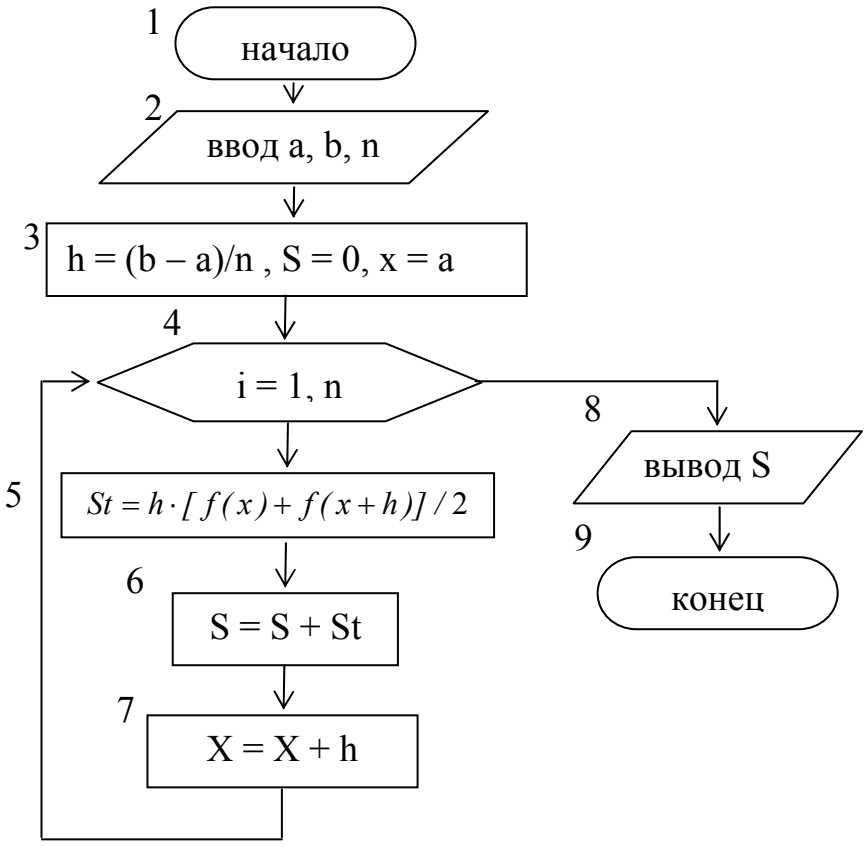

Рисунок 21 Схема алгоритма вычисления интеграла методом трапеций

Программа вычисления $\int_a^b \frac{e^x}{x+1} dx$ методом трапеций, согласно приведённому алгоритму имеет вид.

```
REM Вычисление интегралов методом трапеций
PRINT “Введите A, B, N”
INPUT A, B, N
H = (B-A) / N
S = 0
X = A
FOR  I = 1 TO N
ST = H * (EXP(X) / (X+1) + EXP (X+H) / (X+H+1)) / 2
S = S + ST
X = X + H
NEXT I
PRINT “Интеграл = “ ; S
END
```

Вопросы для самоконтроля

1 В чем сущность численных методов вычисления определенного интеграла?

2 От чего зависит точность вычисления интеграла?

3 В чем отличие метода трапеций от метода прямоугольников?

4 Чем объясняется более высокая точность при вычислении интеграла методом трапеций?

ЗАДАНИЕ 4

Студент должен выполнить индивидуальное задание 4 по своему варианту, номер которого совпадает с индивидуальным номером студента в учебном журнале группы.

Перечень вопросов к заданию 4

Работа в ОС Windows

1 Как создается папка? Как перенести, скопировать файл в папку?

2 Как правильно указать путь доступа к файлу? Объяснить на примере дерева папок.

3 Как скопировать, переместить, удалить, переименовать объект?

4 Какие существуют приемы, чтобы развернуть, закрыть, свернуть, восстановить свернутое окно, изменить размер окна?

5 Как скопировать, переместить и удалить группу файлов?

6 Как сохранить файл под текущим именем или под новым и указать путь доступа к нему?

7 Как использовать буфер обмена?

8 Как восстановить ошибочно удалённый файл? Пояснить основные приёмы работы с Корзиной?

Текстовый редактор Word

9 Как изменить шрифт, его размер, начертание, цвет? Как осуществляется выравнивание текста?

10 Как вставить рисунок в текст?

11 Как выполнить предварительный просмотр текста перед печатью документа?

12 Как скопировать фрагмент текста в другой текст?

13 Как проверить наличие ошибок в документе. Работа с диалоговым окном Правописание?

14 Как создать таблицу с заданным количеством строк и столбцов?

15 Как вставляются и удаляются строки и столбцы таблицы и как изменить их размеры (высоту и ширину)?

16 Как набрать и отредактировать формулу?

17 Как изменить масштаб просмотра документа?

18 Как выделить отдельное слово, целый абзац текста? Как изменить размер, начертание, цвет шрифта?

19 Как осуществляется выравнивание текста и выставить отступ у первой строки абзаца?

20 Как выполнить предварительный просмотр текста перед печатью документа и указать необходимое количество страниц в режиме предварительного просмотра?

21 Как скопировать выделенный фрагмент текста в буфер обмена и как вставить его из буфера обмена в нужное место документа?

Работа в табличном процессоре Excel

22 Какие правила ввода данных в ячейки Excel и их исправления?

23 Как набрать и отредактировать формулу в Excel?

24 Какие бывают типы ссылок на ячейки в Excel и чем они отличаются?

25 Основные виды диаграмм в Excel и способ их получения?

26 Как в Excel скопировать содержимое одной ячейки в другую?

27 Как скопировать, удалить лист или сразу несколько листов из книги?

28 Как с помощью Мастера диаграмм производится оформление диаграммы?

29 Как переименовать лист и вставить новый лист в книгу?

30 Описать 3 способа переименования листа в книге.

ЛИТЕРАТУРА

1 Прищепов, М.А Экзамен по информатике. Основы алгоритмизации и программирования : справ. пособие / М.А. Прищепов, В.П. Степанцов, Е.В. Севернёва. – Мн. : ТетраСистемс, 2001. – 192 с.

2 Прищепов, М.А. Программирование на языках Basic, Pascal и Object Pascal в среде Delphi : учеб. пособие / М.А. Прищепов, Е.В. Севернёва, А.И. Шакирин. – Мн. : ТетраСистемс, 2006. – 320 с.

3 Симонович, С.В. Информатика. Базовый курс: учебное пособие для студентов высших технических учебных заведений / С.В. Симонович [и др.]. – СПб: Питер, 2001. – 640 с.

4 Турчак, Л.И. Основы численных методов : учебное пособие / Л.И. Турчак, П.В. Плотников. – Москва : Физмат-лит, 2002. – 304 с.

СОДЕРЖАНИЕ