## **Тема 2.4. Базовые алгоритмы и примеры их реализации**

[2.4.1. Понятие базовых алгоритмов](#_2.4.1._Понятие_базовых)

[2.4.2. Алгоритмы линейной структуры](#_2.4.2._Алгоритмы_линейной)

[2.4.3. Базовые алгоритмы разветвляющихся структур и примеры их программирования](#_2.4.3._Базовые_)

[2.4.4. Базовые алгоритмы регулярных циклических структур и примеры их программирования](#_2.4.4._Базовые_)

[2.4.5. Базовые алгоритмы итеративных циклических структур и примеры их программирования](#_2.4.5._Базовые_алгоритмы)

[2.4.6. Базовые алгоритмы обработки одномерных массивов](#_2.4.6._Базовые_алгоритмы)

[2.4.7. Базовые алгоритмы обработки двумерных массивов](#_2.4.7._Базовые_алгоритмы)

[2.4.8. Контрольные вопросы по теме «Базовые алгоритмы и примеры их реализации»](#_2.4.8._Контрольные_вопросы)

[2.4.9. Тестовые задания по теме «Базовые алгоритмы и примеры их реализации»](#_2.4.9._Тестовые_задания)

### 2.4.1. Понятие базовых алгоритмов

Базовые алгоритмы обработки данных являются результатом исследований и разработок, проводившихся на протяжении десятков лет. Но они, как и прежде, продолжают играть важную роль в расширяющемся применении вычислительных процессов.

К базовым алгоритмам императивного программирования можно отнести:

* Простейшие алгоритмы, **реализующие базовые алгоритмические структуры.**
* Алгоритмы **работы со структурами данных**. Они определяют базовые принципы и методологию, используемые для реализации, анализа и сравнения алгоритмов. Позволяют получить представление о методах представления данных. К таким структурам относятся связные списки и строки, деревья, абстрактные типы данных, такие как стеки и очереди.
* Алгоритмы **сортировки**, предназначенные для упорядочения массивов и файлов, имеют особую важность. С алгоритмами сортировки связаны, в частности, очереди по приоритету, задачи выбора и слияния.
* Алгоритмы **поиска**, предназначенные для поиска конкретных элементов в больших коллекциях элементов. К ним относятся основные и расширенные методы поиска с использованием деревьев и преобразований цифровых ключей, в том числе деревья цифрового поиска, сбалансированные деревья, хеширование, а также методы, которые подходят для работы с очень крупными файлами.
* **Алгоритмы на графах** полезны при решении ряда сложных и важных задач. Общая стратегия поиска на графах разрабатывается и применяется к фундаментальным задачам связности, в том числе к задаче отыскания кратчайшего пути, построения минимального остовного дерева, к задаче о потоках в сетях и задаче о паросочетаниях. Унифицированный подход к этим алгоритмам показывает, что в их основе лежит одна и та же процедура, и что эта процедура базируется на основном абстрактном типе данных очереди по приоритету.
* Алгоритмы **обработки строк** включают ряд методов обработки (длинных) последователей символов. Поиск в строке приводит к сопоставлению с эталоном, что в свою очередь ведет к синтаксическому анализу. К этому же классу задач можно отнести и технологии сжатия файлов.

Далее рассмотрим алгоритмы, **реализующие базовые алгоритмические структуры.**

### **2.4.2. Алгоритмы линейной структуры**

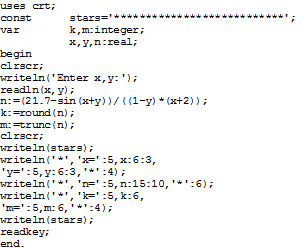
**Пример 2.4.2-1.** Вычислить и вывести на экран значения следующих величин:

****

где x = -1,4; y = 0,8; переменные k и m– целого типа, остальные переменные - вещественного типа; [n]- целая часть числа n.

Схема алгоритма и программы на языках QBasic, Pascal, C++ , представлены на рис. 2.4.2-1.

|  |  |
| --- | --- |
| Пр-3-03-01-Cx.png | Пр-3-03-01-Bas.png |



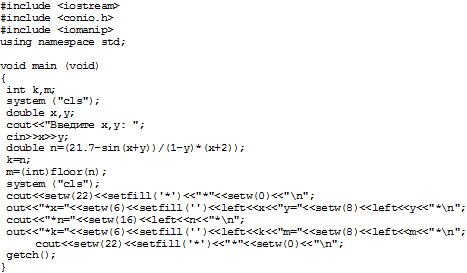
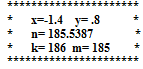


Рис.2.4.2-1

Результаты выполнения программы при указанных выше значе­ниях исходных данных имеют следующий вид:



Следует обратить внимание на то, что целая переменная **k** получила округленное значение **n**, а целая переменная **m** - усе­ченное с помощью функции **FIX( )** до целой части значения **n.**

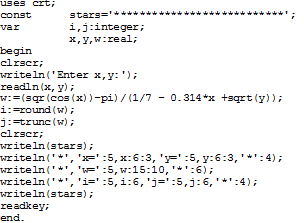
**Пример 2.4.2-2.** Вычислить и вывести на экран значения следующих величин:



где x = 2.9512; y = 0.098633;переменные i и j – целого типа; остальные переменные – вещественного типа.

Схема алгоритма и коды программ представлены на рис. 3.2.1-2.

|  |  |
| --- | --- |
| Пр-3-03-02-Cx.png | Пр-3-03-02-Bas.png |



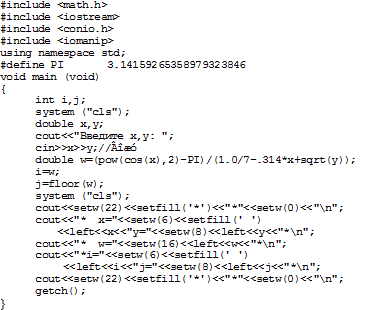
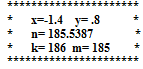


Рис. 2.4.2-2.

Результаты выполнения программы при указанных выше значе­ниях исходных данных имеют следующий вид:

****

**Пример 2.4.2-3.** Вычислить и вывести на экран значение первой космической скорости.

Проведем формализацию. Минимальная скорость, при которой космический аппарат в гравитационном поле Земли может стать искусственным спутником, равна



где  – гравитационная постоянная; M – масса Земли; – расстояние от центра Земли до космического аппарата.

Схема алгоритма и коды программ представлены на рис. 3.2.1-3.

|  |  |
| --- | --- |
| **Пр-3-03-03-Cx.png** | Пр-3-03-03-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-03-Pas.png | Пр-3-03-03-C++.png |

Рис. 2.4.2-3.

Результаты выполнения программы при указанных выше значе­ниях исходных данных имеют следующий вид:

V=7913.047866063509

### 2.4.3. Базовые алгоритмы разветвляющихся структур и примеры их программирования

**Пример 2.4.3-1.** Вывести на экран значение переменной x



Данную задачу можно решить двумя способами. На рис 2.4.3-1а при­ведена схема алгоритма решения задачи, в которой использовано стан­дартное разветвление, а на рис.2.4.3-1б – усеченное разветвление. Второй способ в данном случае предпочтительнее, поскольку упрощает про­граммную реализацию алгоритма. Усеченное разветвление реко­мендуется использовать в тех случаях, когда выбор из двух возможных ветвей производится последовательно и неоднократно.

|  |  |
| --- | --- |
| Пр-3-03-05a-Cx.png | Пр-3-03-05a-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-05a-Pas.png | Пр-3-03-05a-C++.png |

Рис. 2.4.3-1а.

|  |  |
| --- | --- |
| Пр-3-03-05b-Cx.png | Пр-3-03-05b-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-05b-Pas.png | #include <math.h>  #include <iostream>  #include <conio.h>  #include <iomanip>  using namespace std;  void main (void)  ///////////////////////// |

Рис. 2.4.3-1 б.

**Пример 2.4.3-2.** Вычислить значение функции y(x) в соответствии со следую­щим правилом:

****

Схема алгоритма и коды программ вычисления сложной функции приведены на рис.2.4.3-2, где разветвление реализовано с использованием блочного IF.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Пр-3-03-06-Bas.png | |
| Пр-3-03-06-Pas.png | | Пр-3-03-06-C++.png | |

Рис. 2.4.3-2.

**Пример 2.4.3-3.** Вычислить и вывести на экран значение функции

****

Схема алгоритма решения данной задачи (рис. 2.4.3-3) имеет три ветви и тре­бует проверки двух условий. Следует отметить, что в данной схеме алгоритма при выборе второго условия для проверки значения **х** предпочтение отдано не сложному условию   
**(-1< х <1)**, а простому **(х ≥ 1)**.

|  |  |  |
| --- | --- | --- |
| Пр-3-03-07-Cx.png | **Пр-3-03-07-Bas.png** | |
| Пр-3-03-07-Pas.png | | Пр-3-03-07-C++.png | |

Рис. 2.4.3-3.

**Пример 2.4.3-4.** Присвоить переменной f наибольшее из значений двух вво­димых чисел x, y ( f = max{x, y } ).

Алгоритмы и программы решения задачи вы­бора наибольшего из двух значений приведены на рис.2.4.3-4а (стандартное разветвление) и 2.4.3-4б(усеченное разветвление).

|  |  |
| --- | --- |
| Пр-3-03-08a-Cx.png | **Пр-3-03-08a-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-08a-Pas.png | Пр-3-03-08a-C++.png |

Рис. 2.4.3-4а.

|  |  |
| --- | --- |
| Пр-3-03-08b-Cx.png | **Пр-3-03-08b-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-08b-Pas.png | Пр-3-03-08b-C++.png |

Рис. 2.4.3-4b.

В первом способе (рис. 2.4.3-4а) используется стандартное разветвление, а во втором (рис.2.4.3-4б) – усеченное.

**Пример 2.4.3-5.** Вычислить 

Решение задач выбора наибольшего (наименьшего) из значений переменных или выражений числом более двух сводятся к последовательному применению усеченных разветвлений.

Алгоритм, реализующий пример 2.4.3-5 (рис.2.4.3-5), относится к числу базовых алгоритмов выбора наименьшего (наибольшего) из нескольких значений. В основу алгоритма положено усеченное разветвление. Здесь первоначально переменной r присваивается значение первого из выражений данной последовательности. Истинное значение наименьшего значения определяется путем последовательного сравнения со всеми остальными значениями выражений заданной последовательности.

|  |  |
| --- | --- |
| **Пр-3-03-09-Cx.png** | **Пр-3-03-09-Bas.png** |

|  |  |
| --- | --- |
| **Пр-3-03-09-Pas.png** | **Пр-3-03-09-C++.png** |

Рис. 2.4.3-5.

**Пример 2.4.3-6.** Задано число n. Если n – натуральное нечетное число, меньше 10, то целочисленной переменной f присвоить 1, если n – натуральное четное число, то переменной f присвоить 2, если n – отрицательное, то присвоить f значение-1, если значение **n** находится внутри отрезка [10,1000], то f присвоить 10, иначе fприсвоить 0.

Схема алгоритма и коды программ решения задачи 2.4.3-6 с использованием оператора SELECT CASE приведены на рис.2.4.3-6.

|  |  |
| --- | --- |
| Пр-3-03-10-Cx.png | **Пр-3-03-10-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-10-Pas.png | Пр-3-03-10-C++.png |

Рис.2.4.3-6.

### 2.4.4. Базовые алгоритмы регулярных циклических структур и примеры их программирования

**Алгоритмы регулярной циклической структуры** позволяют описать широкий класс задач: вычисление значений функций от одной и более переменных с заданным диапазоном и шагом их изменения; определе­ние наибольшего (наименьшего) значения функции одной и более пере­менных; вычисление конечных сумм и произведений значений величин; ввод, формирование, обработка и вывод элементов массива и другие.

Рассмотрим последовательность данных . В неко­торых случаях член последовательности  можно определить через k предыдущих членов той же последовательности:****

То есть, для нахождения i-го члена последовательности должны быть заданы члены a0,a1,…,ai-1**.** Такое определение называется **ите­рацией** или **рекуррентной формулой**, а алгоритмы вычисления произвольного члена последовательно­сти – **итерационными**. Как пра­вило, итерационные алгоритмы реализуются с помощью циклов – как ре­гулярных, так и итеративных.

В простейшем случае используется выражение  для  где с – известное значение нулевого члена последовательности. Например, для арифметической прогрессии , а для геометрической прогрессии При программировании итераций такого типа используется выражение a = f(a). Использование этого выражения основано на специфике операции присваивания. Сначала вычисляется правая часть выражения при предыдущем значении a**,** равном****, а затем переменной a присваивается вычисленное значение, тем самым получается текущее значение a**,** равное**.**

С помощью рекуррентных формул удобно также находить суммы и про­изведения членов последовательности. Обозначим – сумму k членов последовательности . Очевидно, что  для k = 1,… n; начальное значение суммы следует обнулить:  Теперь найдем произведение k членов этой последовательности по формуле  для k = 1,…,n, для чего начальное значение произведения необходимо установить равным единице, 

Заметим, что с использованием регулярных циклических структур можно находить суммы с конечным числом слагаемых и произведение с конечным, то есть, с заранее заданным числом сомножителей. Вычисление суммы и произведения заранее неизвестного числа членов последовательности осуществляется с помощью итеративных циклов и будет рассмотрено в следующей теме.

В регулярных циклах число повторений должно быть определено заранее. При этом, в одних случаях это число задано явно (константой или вводимым значением переменной), а в других случаях его надо предварительно вычислить.

Количество повторений цикла (n) может быть определено через граничные значения и шаг изменения параметра по формуле:

,

где m1 и m2, соответственно, нижняя и верхняя граница изменения параметра цикла, а h – шаг изменения параметра. Квадратные скобки означают, получение целой части от ре­зультата деления.

Наконец, возможны и такие случаи, когда число повторений цикла не фиксируется в алгоритме в явном виде, а определяется неявно граничными значениями и шагом изменения не­которых переменных (пример 2.4.4-1).

**Пример 2.4.4-1.** Вычислить значение функции y(x) = sin(x) при значе­ниях x, изменяющихся на отрезке [a;b] с шагом h. Другой формулировкой данной задачи может быть следующая задача.

Получить таблицу значений функции y(x)= sin(x) при изме­нении x от a до b с шагом h. Значения a, b, h- вводимые величины.

Схема и коды программ представлены на рис. 2.4.4-1а – 3.1.3-1b.

|  |  |
| --- | --- |
| Пр-3-03-11a-Cx.png | **Пр-3-03-11a-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-11a-Pas.png | Пр-3-03-11a-C++.png |

Рис. 2.4.4-1a

В алгоритме решения данной задачи (Рис.2.4.4-1а) используется регулярная циклическая структура, которая в программе реализована оператором FOR…NEXT**,** а в качестве параметра цикла используется вещественный аргумент функции **-** х.

 Такой способ организации цикла опасен тем, что в результате погрешностей округления последнее зна­чение параметра может не оказаться равным в точности значению верхней границы b, и если оно окажется больше b**,** то выполнение цикла прекратится, aв результате таблица значений функции будет содержать на одну точку меньше. Так, например, если a=0, b=1 и h=0.1, то последнее? выведенное в цикле значение x оказывается равным 0.9000001, то есть точка x=b в таблицу не попадет. Такой способ организации цикла в данной задаче можно использовать только, если переменные a, b, h и x целого типа.

Избежать подобных неприятностей можно путем предварительного вычисления числа повторений цикла и использования в качестве параметра цикла **переменной целого типа**, назначение которого – подсчет количества повторений цикла (рис. 2.4.4-1b и 2.4.4-1c). Различие между алгоритмами состоит только в способах вычисления текущего значения аргумента x. В первом алгоритме, очередное значение **х** вычисляется через предыдущее, путем добавления шага, а во втором алгоритме **х** вычисляется через параметр цикла.

|  |  |
| --- | --- |
| Пр-3-03-11b-Cx.png | **Пр-3-03-11b-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-11b-Pas.png | Пр-3-03-11b-C++.png |

Рис. 2.4.4-1b.

|  |  |  |
| --- | --- | --- |
| Пр-3-03-11c-Cx.png | **Пр-3-03-11c-Bas.png** | |
| Пр-3-03-11c-Pas.png | | Пр-3-03-11c-C++.png | |

Рис. 2.4.4-1c.

**Пример 2.4.4-2.** Вычислить сумму положительных значений функции y=cos(8x) при изменении x от 0 до π с шагом π/16.

Алгоритм решения данной задачи относится к алгоритмам вычисления конечных сумм (рис.2.4.4-2).

|  |  |
| --- | --- |
| Пр-3-03-12-Cx.png | Пр-3-03-12-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-12-Pas.png | Пр-3-03-12-C++.png |

Рис. 2.4.4-2

Алгоритм использует цикл с параметром целого типа. Для уменьшения влияния ошибок округления при таком спо­собе вычисления текущего значения аргумента используется представление вещественных чисел с удвоенной точностью. Количество повторений цикла можно вычислить заранее по вышеприведенной формуле или непосредственно программой перед входом в цикл .

Так как при таком способе организации цикла (через целый параметр), аргумент x уже не будет автоматически изменяться при каждом повторении цикла, то в тело цикла добавляется оператор x=x+h, который обеспечивает изменение **x** на величину шага h при каждом повторении цикла. При этом начальное значение аргумента x должно быть установлено перед первым входом в цикл (x=0). Начальное значение суммы также следует установить равным нулю до цикла (s=0), чтобы после нахождения первого подходящего значения y>0, значение суммы стало равно этому значению. Заметим, что если в задаче требуется вычислить конечное произведение, то начальное значение переменной, которая будет накапливать произведение, следует установить равным единице (p=1).

**Пример 2.4.4-3.** Вычислить сумму и произведение **n** членов последовательности ****

Схема алгоритма и коды программ приведены на рис. 2.4.4-3.

|  |  |
| --- | --- |
| **Пр-3-03-13-Cx.png** | Пр-3-03-13-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-13-Pas.png | Пр-3-03-13-Pas.png |

Рис. 2.4.4-3.

**Пример 2.4.4-4.** Вычислить **n**-ый член последовательности 

В данном примере каждый очередной член последовательности зависит от двух предыдущих. Обозначим Тогда очередной член последовательности a=b+c**.** Операторы b=c и c=a изменяют значения b и c для очередного вычисления a. Таким образом, значение предыдущего члена последовательности присваивается предыдущему члену последовательности (b=c), то есть предшествующему этому, а значение только что вычисленного члена **a** присваиваем предыдущему **(**c=a**).** Начальное значение параметра цикла i=3, так как вычисление начинается с 3-его члена последовательности (f0=f1=**1** по условию задачи).

Схема алгоритма и коды программ приведены на рис. 2.4.4-4.

|  |  |
| --- | --- |
| Пр-3-03-14-Cx.png | Пр-3-03-14-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-14-Pas.png | Пр-3-03-14-C++.png |

Рис. 2.4.4-4.

**Пример 2.4.4-5.** Вычислить значение факториала заданного числа n.

По определению факториал

f = n! = 1∙ 2 ∙ 3 ∙ … ∙ (n-1) ∙ n (при n≥1) и f=0! = 1(при n=0).

Таким образом, задача вычисления факториала сводится к задаче вычисления произведения значений целых чисел, изменяющихся от 1 до n с шагом 1.

Накопление произведения будем производить по рекуррентной формуле: 

Схема алгоритма и программа приведены на рис. 2.4.4-5.

|  |  |
| --- | --- |
| **Пр-3-03-15-Cx.png** | Пр-3-03-15-Bas.png |

|  |  |
| --- | --- |
| **Пр-3-03-15-Pas.png** | **Пр-3-03-15-C++.png** |

Рис. 2.4.4-5.

**Пример 2.4.4-6.** Найти наименьшее значение функции



где c, d – входные параметры, а **x**изменяется на отрезке [a;b]с шагом h.

Алгоритм нахождения наименьшего (наибольшего) значения функции основан на последовательном сравнении очередного значения функции с текущим наименьшим (наибольшим). В качестве начального наименьшего значения ymin следует использовать число, близкое к наибольшему числу из диапазона возможных значений того типа данных, который описывает рассматриваемую функцию, чтобы при первом проходе цикла наверняка выполнилось условие y<ymin. Здесь за ymin можно принять число 1038**,**  близкое к наибольшему из диапазона значений типа SINGLE.

Соответственно, за начальное значение **ymax** можно принять число -1038, то есть число, близкое к наименьшему числу из диапазона значений SINGLE, для того, чтобы при первом проходе цикла гарантированно выполнялось условие y>ymax. Для определения значения аргумента xmin (xmax), соответствующего найденному наименьшему (наибольшему) значению функции при каждом переопределении текущего наименьшего (наибольшего) значения функции необходимо одновременно переопределять текущее значение xmin (xmax).

Схема алгоритма и программа приведены на рис. 2.4.4-6.

|  |  |
| --- | --- |
|  | Пр-3-03-16-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-16-Pas.png | Пр-3-03-16-C++.png |

Рис. 2.4.4-6.

**Пример 2.4.4-7.** Получить таблицу значений функции в заданных диапазонах значений аргументов, если шаг изменения x равен , а шаг изменения **y** равен **.**

Схема алгоритма и код программы приведены на рис. 2.4.4-7.

Получение таблицы значений функции от двух параметров реализуется алгоритмом, использующим вложенные циклы. Во внешнем цикле происходит перебор всех значений переменной y, а во внутреннем, при каждом фиксированном значении переменной **у,** происходит полный перебор значений параметра **х**.

В качестве параметров циклов используются переменные целого типа, выполняющие роль счетчиков значений изменяющихся параметров.

|  |  |
| --- | --- |
| Пр-3-03-17-Cx.png | Пр-3-03-17-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-17-Pas.png | Пр-3-03-17-C++.png |

Рис. 2.4.4-7.

### 2.4.5. Базовые алгоритмы итеративных циклических структур и примеры их программирования

При описании базовых алгоритмов регулярных циклических структур мы приводили примеры вычисления сумм и произведений конечного (заранее известного) числа членов последовательности (примеры 2.4.4-2, 2.4.4-3, 2.4.4-5).

Рассмотрим теперь некоторую последовательность, содержащую бесконечное число членов: В данном случае требуется вычислять члены последовательности до тех пор, пока очередной вычисленный член не будет удовлетворять некоторому условию.

Если задано некоторое число х**,** условия окончания итерационного процесса могут, например, быть следующими:

* для убывающей последовательности 
* для возрастающей последовательности  **(**примеры 2.4.5-1, 2.4.5-3, 2.4.5-5**)**;
* для убывающей знакопеременной последовательности | **(**пример2.4.5-4);
* для некоторых других последовательностей ****(пример 2.4.5-2).

Рассмотрим примеры алгоритмов итерационного типа, реализуемых с использованием итеративных циклов.

**Пример 2.4.5-1.** Среди чисел  найти первое число, большее вводимого значения переменной a.

Алгоритм решения данной задачи относится к алгоритмам вычисления членов бесконечных последовательностей (Рис.2.3.5-1a). Он использует итеративный цикл с предусловием и про­граммируется с помощью оператора DO WHILE ... LOOP.

Для решения данной задачи проведем формализацию. Для этого введем следующие обозначения: b – очередной член бесконечной последовательности; n – номер этого члена, который в данной задаче совпадает со значением знаменателя дроби, добавляемой к предыдущему члену для получения значения очередного члена последовательности. Таким образом, итерационная формула вычисления очередного члена последовательности

, где n – номер члена.

|  |  |
| --- | --- |
| Пр-3-03-18-Cx.png | Пр-3-03-18-Bas.png |

|  |  |
| --- | --- |
| **Пр-3-03-18-Pas.png** | Пр-3-03-18-C++.png |

Рис. 2.4.5-1a.

Решение данного примера может быть реализовано с использованием конструкции DOUNTIL…LOOP(рис. 2.4.5-1b), а также цикл с предусловием можно заменить на цикл с постусловием с соответствующим изменением настройки цикла (рис. 2.4.5-1c).

Изменим в цикле рис. 2.4.5-1а условие продолжения выполнения цикла, то есть будем проводить вычисления членов последовательности до тех пор, пока не встретится член, больший заданного числа. Поэтому, заменим условие b<=a на условие b>a.

Тогда алгоритм и программа будет выглядеть так, как на рис. 2.4.5-1b.

|  |  |
| --- | --- |
| Пр-3-03-18b-Cx.png | Пр-3-03-18b-Bas.png    . |

|  |  |
| --- | --- |
| Пр-3-03-18b-Pas.png |  |

|  |  |
| --- | --- |
| Пр-3-03-18b-C++.png |  |

Рис.2.4.5-1b

Если вычисление всех членов последовательности, начиная с первого, проводится в теле цикла, то алгоритм и программа принимают вид, показанный на рис 2.4.5-1с.

|  |  |  |
| --- | --- | --- |
| Пр-3-03-18c-Cx.png | Пр-3-03-18c-Bas.png | |
| Пр-3-03-18c-Pas.png | |  | |

|  |  |
| --- | --- |
| Пр-3-03-18c-C++.png |  |

Рис. 2.4.5-1с.

В этом случае мы получили структуру итеративного цикла с постусловием, изменив только настройку цикла: n=0, b=0. Здесь условием окончания вычислительного процесса служит значение “Истина” выражения b>a, а продолжением – значение “Ложь” (конструкция DO…LOOP UNTIL).

Изменив условие на b<=a**,** получим конструкцию цикла с постусловием DO…LOOP WHILE**.**

Следовательно, задача может быть решена с использованием как цикла с предусловием, так и с постусловием, при этом с любой из конструкций WHILE или UNTIL**.**

**Пример 2.4.5-2.** Вычислить с точностью **** корень уравнения   
**ex + x = 0**, вос­пользовавшись итерационной формулой **, ** Закончить итеративный процесс, как только станет меньше **.**

Для решения этой задачи необходимо из очередного вычисленного корня  вычитать предыдущий корень **.** Для этого, при каждом повторении цикла перед вычислением очередного корня x**,**  следует сохранить в переменной a текущее значение x (оно становится предыдущим). Цикл прекращаем, если разность между a (то есть ****) и x (то есть ) станет меньше ****.

Алгоритм решения данной задачи относится к итерационным алгоритмам (рис.2.4.5-2) и может быть реализован с помощью оператора DO ... LOOP UNTIL c постусловием.

|  |  |
| --- | --- |
| Пр-3-03-19-Cx.png | Пр-3-03-19-Bas.png |

|  |  |
| --- | --- |
| **Пр-3-03-19-Pas.png** | **Пр-3-03-19-C++.png** |

Рис.2.4.5-2.

**Пример 2.4.5-3**. Задана бесконечно возрастающая последовательность



Требуется вычислять все члены последовательности, до тех пор, пока значение очередного члена не превысит некоторое число d (3<d<1000).

В нашей задаче для вычисления любого члена последовательности можно воспользоваться формулой

,где n=0, 1, 2,… - номер члена последовательности.

В приведенном алгоритме (рис. 2.4.5-3) имеется проверка правильности ввода исходных данных (x и d). Кроме того, во многих задачах, использующих итеративные алгоритмические структуры, следует предусмотреть так называемую “страховку от зацикливания”, так как иногда условие продолжения цикла остается истинным бесконечно. В данном примере цикл с постусловием будет выполняться не более 100 раз, даже если очередной член последовательности будет оставаться меньше d.

|  |  |
| --- | --- |
| Пр-3-03-20-Cx.png | **Пр-3-03-20-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-20-Pas.png  Рис. 2.4.5-3. | Пр-3-03-20-C++.png |

**Пример 2.4.5-4.** Вычислить сумму членов знакопеременной убывающей последовательности с заданной точностью ε**:**



Вычисление с заданной точностью **ε** означает, что суммирование членов ряда надо продолжать до тех пор, пока очередной вычисленный член ряда не станет меньше по абсолютной величине числа ε.

Отметим, что во многих задачах непосредственный подсчет очередного члена связан с вычислительными трудностями. В нашем примере целесообразно использовать рекуррентную формулу, которая позволяет вычислить значение переменной на следующем шаге, используя ее значение на текущем шаге - . Выражение для q можно получить, разделив  член на член.

Приведем вывод рекуррентной формулы для заданного в задании ряда. Формула n-го члена



тогда формула n+1члена



Разделив член на , получим выражение для q



Таким образом, рекуррентная формула для данного ряда:



Выбор начального значения номера члена ряда (**n**) для нашего случая будет n=0, так как при подстановке этого значения в формулу **n**-го члена ряда



мы получим значение первого члена, равного x-1илиa = x - 1.

Схема алгоритма и код программы приведены на рис. 2.4.5-4a. Этот алгоритм относится к алгоритму итеративного цикла с предусловием, в котором, чтобы избежать возможного зацикливания предусмотрено выполнение цикла не более 100 раз.

|  |  |
| --- | --- |
| Пр-3-03-21a-Cx.png | **Пр-3-03-21a-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-21a-Pas.png | Пр-3-03-21a-C++.png |

Рис. 2.4.5-4а.

Рекуррентную формулу для вычисления очередного значения члена последовательности, приведенную выше, можно вычислить по-другому, используя его значение на предыдущем шаге 

Для этого случая вывод будет следующий:



.

Тогда

.

Таким образом, рекуррентная формула в этом случае:

.

Схема алгоритма и код программы приведены на рис. 2.4.5-4b. В этом случае, в отличие от предыдущего, увеличивать n (n=n+1) следует до, а не после вычисления очередного члена ряда.

|  |  |
| --- | --- |
| Пр-3-03-21b-Cx.png | Пр-3-03-21b-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-21b-Bas.png |  |

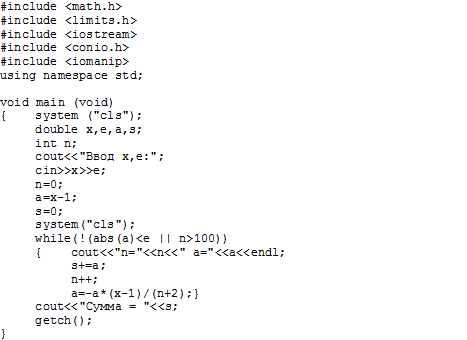


Рис. 2.4.5-4b.

**Пример 2.4.5-5.**Вычислить сумму всех чисел Фибоначчи, которые не

превосходят заданного натурального числа m.

Числа Фибоначчи (**** ) определяются по формулам:

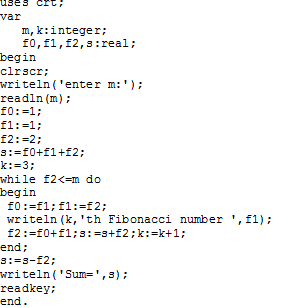
****при i = 2, 3,...,

то есть каждое очередное число равно сумме двух предыдущих.

Подобная задача была решена при рассмотрении регулярных циклических структур (пример 2.4.4-4). Однако, здесь число слагаемых (членов последовательности) заранее неизвестно, поэтому организуем итеративный цикл с предусловием. В данном примере до начала цикла задаются значения трех чисел Фибоначчи **** и и их сумма . В цикле происходит переопределение значений переменных ****, вывод очередного (предыдущего) числа Фибоначчи, вычисление следующего числа с последующим добавлением его в сумму. Когда текущее число Фибоначчи превосходит заданное число m (- ложно), то происходит выход из цикла. Поскольку последнее число Фибоначчи уже добавлено в сумму, то после выхода из цикла его необходимо из нее вычесть (). Таким образом, в окончательном значении суммы будут учтены только те числа Фибоначчи, которые еще не превосходят заданное число m.

Схема алгоритма и коды программ приведены на рис. 2.4.5-5

|  |  |
| --- | --- |
| Пр-3-03-23-Cx.png | **Пр-3-03-23-Bas.png** |



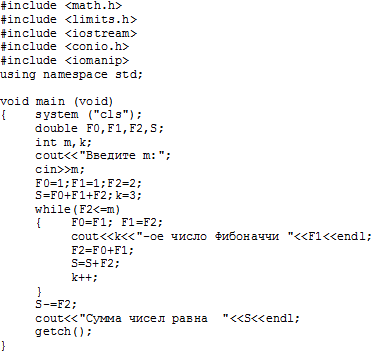


Рис. 2.4.5-5.

### 2.4.6. Базовые алгоритмы обработки одномерных массивов

Для работы с массивами, как правило, используются алгоритмы регулярной циклической структуры. В примерах 2.4.6-1 – 2.4.6-9 приведены алгоритмы и программы, использующие регулярные циклические структуры.

**Пример 2.4.6-1.** Ввести массив a(6), состоящий из 7 вещественных чисел, и вычислить произведение ненулевых элементов этого массива.

В программе, приведенной на рис. 2.4.6-1, объявлен вещественный массив a(6) с элементами а(0), а(1), ... , а(6). Поочередный ввод элементов массива выполняется в цикле. С помощью оператора PRINT на экран выводится запрос с указанием номера вводимого элемента, затем в операторе INPUT присваиваются вводимые числовые зна­чения соответствующим элементам массива a(6). Вывод на экран значений элементов массива **a** также выпол­няется с помощью регулярного цикла, что позволяет визуально про­контролировать результаты ввода. Вывод массива производится в одной строке. При этом каждый элемент мас­сива выводится с указанием имени массива и значения индекса (например: a(2)=12.5), между собой элементы разделяются тремя пробе­лами.

Элементы массива последовательно выводятся в строку благодаря символу **'';''** в конце оператора PRINT. Напомним, что пустой оператор PRINT**,** выполняемый после выхода из этого цикла, прерывает последовательный вывод в строку и устанавливает курсор в начало следующей строки.

|  |  |
| --- | --- |
| Пр-3-03-24-Cx.png | **Пр-3-03-24-Bas.png** |

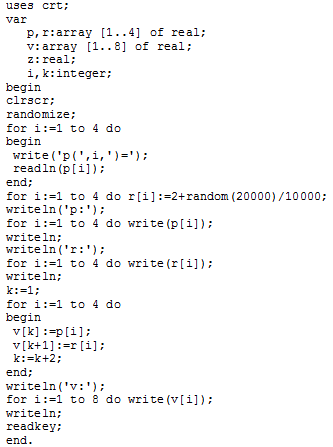
|  |  |
| --- | --- |
| Пр-3-03-24-Pas.png | Пр-3-03-24-C++.png |

Рис. 2.4.6-1

**Пример 2.4.6-2**. Требуется из двух массивов p и r**,** каждый из которых состоит из 4-х элементов, получить третий массив v(1:8), путем последовательного попарного переписывания в него элементов массивов p и r. Элементы массива p(1:4) необходимо ввести с клавиатуры, а элементы массива r(1:4) сформировать как случайные числа, равномерно распределенные на отрезке **[2;4].**

Алгоритм и программа решения данной задачи приведены на рис. 2.4.6-2.

|  |  |
| --- | --- |
| Пр-3-03-25-Cx.png | Пр-3-03-25-Bas.png |



|  |  |
| --- | --- |
|  | Пр-3-03-25-C++.png |

Рис. 2.4.6-2.

При решении данной задачи исходный массив p вводится, а массив r формируется из случайных чисел, равномерно распределенных на заданном отрезке [a;b] по формуле r(i)=a+RND(1)\*(b-a). Случайные числа для заполнения массива r вычисляются с помощью функции RND(1), которая возвращает случайные значения, равномерно распределенные на отрезке [0;1]. Для того, чтобы функция RND(1) возвращала различные значения, необходимо в текст программы включить оператор RANDOMIZETIMER. Умножая значение RND(1) на длину заданного отрезка и прибавляя значение его левой границы, получаем случайные числа, равномерно распреде­ленные на заданном отрезке.

Для формирования массива v используется переменная k - но­мер очередного элемента массива v. В цикле одновременно заполняются два элемента массива v: в элемент с номером k переписывается i-й элемент из массива p, а в элемент с но­мером (k+1) переписывается i-й элемент из массива r. Вывод сформированного массива v выполняется в цикле.

**Пример 2.4.6-3.** Сформировать массив c(9), содержащий 10 элементов, по следующему пра­вилу:

****

В сформированном массиве переставить элементы с целыми значениями в начало массива.

Алгоритм и коды программ решения данной задачи приведены на рис. 2.4.6-3

Для того, чтобы переписать це­лые элементы в начало массива, в переменной k хра­нится номер элемента, в который переписывается очередное целое значение. Чтобы определить, является ли очередной элемент массива целым, проведем сравнение разности значения целой части очередного элемента и значения очередного элемента массива c(i) с нулем. Целая часть значения c(i)выделяется с помощью функции FIX(). Если очередной элемент c(i) содержит целое значение, то произво­дится обмен значений двух элементов массива c(k) и c(i)c помощью переменнойtemp**.** Сформированный массив c выводится на экран в строку с указанием имени массива и номера элемента.

|  |  |
| --- | --- |
| Пр-3-03-26-Cx.png | **Пр-3-03-26-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-26-Pas.png | Пр-3-03-26-C++.png |

Рис. 2.4.6-3.

**Пример 2.4.6-4**. Вычислить произведение положительных и сумму неположительных элементов массива x(1:20).

Алгоритм и программа решения задачи приведены на рис. 2.4.6-4

Накопление суммы и произведения в алгоритме осуществлено с использованием рекуррентных формул: 

Для ввода элементов массива и вычисления их конечной суммы **s** и произведения P используется регулярная циклическая структура, в кото­рой в качестве параметра цикла выступает индекс массива (i = 1,2,…,20).

В данном примере ввод и обработка элементов массива осуществляется в одном цикле.

|  |  |  |
| --- | --- | --- |
| **Пр-3-03-27-Cx.png** | Пр-3-03-27-Bas.png | |
| **Пр-3-03-27-Pas.png** | | | **Пр-3-03-27-C++.png** | |

Рис. 2.4.6-4.

**Пример 2.4.6-5.** Сформировать и вывести на экран массив y(1:n)**,** переписав в него положительные элементы массива x(1:100).

Алгоритм и программа решения задачи приведены на рис. 2.4.6-5.

Для решения поставленной задачи необходимо проверить знак у всех элементов массива х**.** При этом текущий индекс (n) формируемого массива **y** меняется независимо от индекса (i) массива х, а в зависимости от конкретного содержимого элемента x. Индекс n увеличивается на **1** только по мере появления положительного элемента массива х. Таким образом, после проверки всех 100 элементов массива x, в переменной n будет содержаться число положительных элементов массива x, что соответствует количеству элементов массива **y**и используется при его выводе.

|  |  |  |  |
| --- | --- | --- | --- |
| Пр-3-03-28-Cx.png | | **Пр-3-03-28-Bas.png** | |
| Пр-3-03-28-Pas.png | | Пр-3-03-28-C++.png | |

Рис. 2.4.6-5.

При обработке структурированных данных возникает не­обходимость упорядочивания их значений по возрастанию или по убы­ванию. К настоящему времени разработано множество различных **алго­ритмов упорядочения** (сортировки), имеющих свои преимущества и недостатки. Рас­смотрим два наиболее простых и часто используемых алгоритма. При этом, следует иметь в виду, что алгоритмы упорядочения по убыванию представляют собой как бы “зеркальное отображение“ соответствующих алгоритмов упорядочения по возрастанию и отличаются только знаком при сравнении значений 2-х данных.

Отметим, что в схемах алгоритмов, приведенных на рис. 2.4.6-2 – 2.4.6-5 детально показаны фрагменты алгоритмов, в которых осуществляется ввод / вывод элементов одномерных массивов. Такое описание полностью соответствует тексту программы. Для краткости в дальнейших примерах во всех схемах алгоритмов задач ввод / вывод элементов массива будем укрупнено изображать одним блоком. То есть, в блоке будут указываться только имя массива и диапазон индексов вводимых или выводимых элементов массива. Однако, следует помнить, что в программе этот укрупненный блок должен быть представлен соответствующим циклом, предназначенным для ввода или вывода данных.

**Пример 2.4.6-6**. Произвести сортировку по убыванию массива х(1:n), состоящего из **n** элементов, применяя “**сортировку выбором**”.

В программе данного примера (рис.2.4.6-6) показано использование динамического массива х(1:n). Динамический массив объявляется после ввода количества его элементов (n**),** а значения элементов массива можно вводить после объявления массива. Сортировка элементов массива по убыванию производится с помощью вложенных циклов. Суть метода “сортировки выбором” состоит в следующем.

Сначала осуществляется поиск наибольшего элемента массива и его индекса среди всех элементов, начиная с 1-го. Найденный максимальный элемент меняется с первым элементом. Затем вновь осуществляется поиск наибольшего элемента массива и его индекса, но уже со второго элемента, и найденный максимальный элемент обменивается со 2-м элементом и так далее. Таким образом, число найденных максимумов (поисков) равно n-1. Т.е. последний (n-ый) элемент массива встанет на свое место автоматически. Во внешнем цикле, начиная с первого и до предпоследнего элемента, сначала очередной элемент принимается за максимальный, а затем (после выполнения внутреннего цикла) обеспечи­вается заполнение очередного элемента массива наибольшим “среди оставшихся“.

Внут­ренний цикл осуществляет перебор и сравнение последующих эле­ментов, начиная с **(**i+1**)-**го до последнего элемента массива. В результате выполнения внутреннего цикла, в переменной xmax фиксируется значение наибольшего элемента, а в переменной m - его номер. Далее, во внешнем цикле выполняется перестановка найденного максимального эле­мента на место очередного i-го элемента массива.

|  |  |
| --- | --- |
| Пр-3-03-29-Cx.png | **Пр-3-03-29-Bas.png** |

|  |  |
| --- | --- |
| Пр-3-03-29-Pas.png | Пр-3-03-29-C++.png |

Рис. 2.4.6-6.

**Пример 2.4.6-7.** Упорядочить по возрастанию и вывести на экран массив х(1:n) посредством модифицированного метода “пузырька“.

Алгоритм и программа упорядочения массива, по методу «пузырька» приведены на рис.2.4.6-7.

Суть метода “пузырька“ заключается в последовательном сравнении первого элемента массива со вторым, третьим и так далее. Если значения в паре расположены в порядке возрастания, то их меняют местами. После первого просмотра элемент массива с наименьшим значением становится первым. Далее сравниваются все элементы, начиная с третьего, со вторым, в результате чего элемент со значением, следующим за минимальным элементом, становится на второе место.

Продолжая таким же образом, вплоть до предпоследнего элемента, сортируют весь массив.

В QB обмен значениями переменных в программе про­изводится с использованием функции SWAP a, b , где a и b - переменные или элементы массивов, которые меняются своими значениями. Обратите внимание, что в схемах алгоритмов обмен значениями обозначается символом “**↔**”.

|  |  |
| --- | --- |
| Пр-3-03-30-Cx.png | Пр-3-03-30-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-30-Pas.png | Пр-3-03-30-C++.png |

Рис. 2.4.6-7

**Пример 2.4.6-8.** Удалить из массива х(1:n) отрицательные эле­менты.

Алгоритм, реализующий этот процесс, называется алгоритмом “**сжатия**“ (рис.2.4.6-8). Метод заключается в поиске удаляемого элемента, фиксации его номера (например, i-й), а затем последовательной перезаписи всех последующих элементов массива так, что значение следующего i+1 элемента записывается на место i-го , на место (**i+1)-**го элемента – (i+2**)-**й и так далее до конца массива. Из последовательности исчезает значение удаляемого элемента, однако последний элемент повторяется дважды, поэтому после выхода из цикла по перезаписи элементов длина массива должна быть уменьшена на **1**.

Так как отрицательные элементы могут идти подряд, то i+1 элемент, который встал на место i-го тоже может быть меньше 0. Поэтому, перед тем, как вернуться в цикл по параметру i (то есть увеличить i на 1), необходимо снова проверить текущий **i**-й элемент (бывший i+1) и, возможно, тоже удалить его “сжатием”. Возврат во внешний цикл по параметру i можно осуществлять только тогда, когда очередной проверяемый элемент не будет меньше 0.

|  |  |
| --- | --- |
| Пр-3-03-31-Cx.png | **Пр-3-03-31-Bas.png** |

|  |  |
| --- | --- |
| **Пр-3-03-31-Pas.png** | **Пр-3-03-31-C++.png** |

Рис. 2.4.6-8

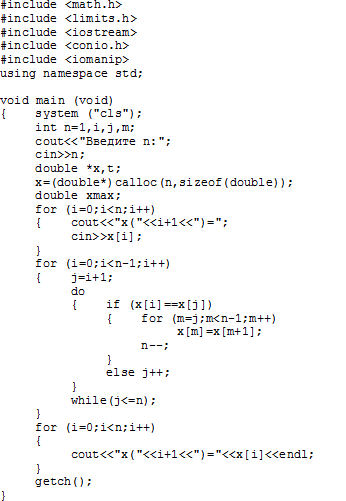
**Пример 2.4.6-9.** Удалить из массива x(1:n) одинаковые элементы.

Алгоритм «сжатия» для получения массива с уникальными элементами и программа, решающая настоящую задачу, приведены на рис. 2.4.6-9.

Метод заключается в последовательном сравнении каждого элемента исходного массива со всеми остальными. Если x(i)=x(j)**,** то j-й элемент удаляется, а длина массива уменьшается на единицу.

|  |  |
| --- | --- |
| Пр-3-03-32-Cx.png | Пр-3-03-32-Bas.png |

|  |  |
| --- | --- |
| **Пр-3-03-32-Pas.png** |  |

****

|  |  |
| --- | --- |
| **Пр-3-03-32-C++.png** |  |

Рис. 2.4.6-9

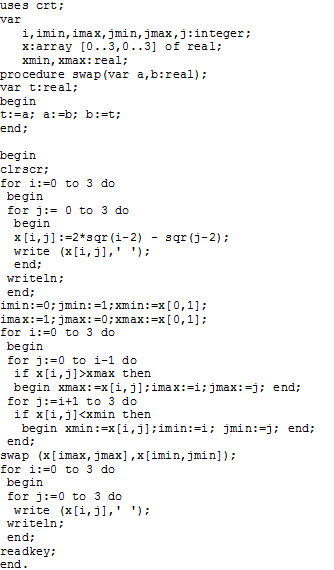
### 2.4.7. Базовые алгоритмы обработки двумерных массивов

Для работы с массивами, чаще всего, используются алгоритмы регулярной циклической структуры. В примерах 2.4.7-1 – 2.4.7-5 приведены алгоритмы, использующие регулярные циклические структуры.

**Пример 2.4.7-1.** Требуется сформировать двумерный массив **x**, состоя­щий из **4** строк и **4** столбцов по правилу: , и определить максимальный элемент среди элементов, лежащих ниже главной диагонали, и минимальный элемент - выше главной ди­агонали. Найденные максимальный и минимальный элементы поме­нять местами.

Алгоритм и программа решения данной задачи представлены на рис. 2.4.7-1.

|  |  |
| --- | --- |
| Пр-3-03-33-Cx.png | **Пр-3-03-33-Bas.png** |

****

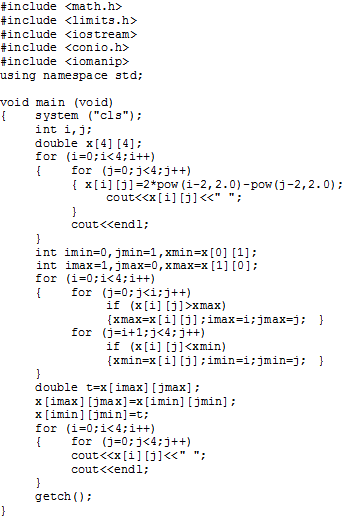
****

Рис. 2.4.7-1

Формирование значений элементов двумерного массива **x** по заданной формуле и их вывод в виде матрицы производится с помощью вложенных циклов. В переменных xmin и xmaxв цикле фиксируются минимальный и мак­симальный элементы массива, в переменных imin , jmin и imax, jmax - их координаты (номер строки и номер столбца), в начале это 1-ый элемент ниже диагонали и 1-ый элемент выше диагонали.

Изменяя номера строки iс помощью внешнего цикла, отдельно просматриваются элементы, расположенные ниже (j от 0 до i-1) и выше (j от i+1 до 3) главной диагонали. После просмотра всех элементов массива **x,** в переменных imaxи jmax фиксируются координаты максимального элемента, а в переменных imin и jmin - координаты минимального элемента.

**Пример 2.4.7-2.** Заполнить массив f(5x2) целыми числами так, чтобы получилась следующая таблица:

1 7 13

2 8 14

3 9 15

4 10 6

5 11 17

6 12 18,

а затем осуществить перестановку элементов первого и последнего столбцов.

Алгоритм и коды программ данной задачи приведены на рис. 2.4.7-2.

|  |  |
| --- | --- |
| Пр-3-03-34-Cx.png | Пр-3-03-34-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-34-Pas.png | Пр-3-03-34-C++.png |

Рис. 2.4.7-2

Данный пример демонстрирует заполнение массива целых чисел f по столбцам и перестановку элементов первого и последнего столбцов. Вложенные циклы обеспечивают перебор элемен­тов двумерного массива по столбцам: внутренний цикл по пер­вому индексу, а внешний по второму индексу. Массив f после заполнения выводится на экран в виде требуемой таблицы.

Перестановка элементов первого и последнего столбцов в программе вы­полняется в цикле, изменяющем номер строки i**,** в котором выполняется обмен значениями элементов f(i,0) и f(i,2). После этого мас­сив выводится на экран в следующем виде:

13 7 1

14 8 2

15 9 3

16 10 4

17 11 5

18 12 6.

**Пример 2.4.7-3.** Сформировать одномерный массив а(1:25**)**, каждый элемент которого представляет собой количество положительных элементов соответствующего столбца двумерного массива b(1:15, 1:25)**.**

Алгоритм и коды программ представлены на рис. 2.4.7-3

|  |  |
| --- | --- |
| Пр-3-03-35-Cx.png | Пр-3-03-35-Bas.png |

.

|  |  |
| --- | --- |
| Пр-3-03-35-Pas.png | Пр-3-03-35-C++.png |

Рис. 2.4.7-3

Для перебора всех элементов исходной таблицы b ис­пользованы вложенные циклы. Во внешнем цикле меняется но­мер столбца, а во внутреннем – номер строки. Такой порядок перебора элементов двумерного массива диктуется условиями задачи. В каждом столбце про­изводится подсчет положительных элементов, и по окончании цикла ре­зультат записывается в соответствующий элемент одномерного массива а, номер ко­торого совпадает с номером столбца матрицы b.

Вывод массива а в ал­горитме показан одним блоком (укрупнено), однако при программиро­вании организованы вложенные циклы.

**Пример 2.4.7-4.** Из матрицы **х** размером 10х15 получить и вывести на экран транспонированную к ней матрицу y(1:15, 1:10).

Алгоритм и коды программ решения данной задачи приведены на рис. 2.4.7-4.

Для решения поставленной задачи в алгоритме трижды необходимо реализовать вложенные циклы: для ввода, обработки и вывода всех элементов исходной матрицы **х**. Транспонирование матрицы осуществляется путем перестановки (обмена) индексов транспонированной матрицы по отношению к исходной. Для краткости, как ввод исходной матрицы, так и вывод матрицы-результата, в схеме алгоритма представлены, укрупнено – одним блоком.

|  |  |
| --- | --- |
| Пр-3-03-36-Cx.png | Пр-3-03-36-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-36-Pas.png | Пр-3-03-36-C++.png |

Рис. 2.4.7-4

**Пример 2.4.7-5.** Получить и вывести на экран алгебраическое произ­ведение р

матрицы хразмером15х20 на матрицу **y** размером 20х10.

Алгоритм и программа решения данной задачи представлены на рис. 2.4.7-5.

|  |  |
| --- | --- |
| Пр-3-03-37-Cx.png | Пр-3-03-37-Bas.png |

|  |  |
| --- | --- |
| Пр-3-03-37-Pas.png | Пр-3-03-37-C++.png |

Рис. 2.4.7-5.

Произведением p матрицы x(m , n) на матрицу y (n , k) называется матрица (m , k), в которой элемент, стоящий на пересечении i-й строки и j-ого столбца, равен произведению i-го вектора-строки матрицы xна j-й вектор-столбец матрицы y**.** А произведением вектора-строки a на вектор-столбец b с одинаковым числом элементов n называется сумма произведений соответствующих элементов этих векторов

.

Произведение двух матриц можно записать следующим образом:



Из опреде­ления следует, что в нашем случае матрица p будет иметь размер 15х10. Для формирования такой матрицы необходимо организовать вложенные циклы: внешний цикл, изменяющий номер строки от 1 до 15, и внутренний цикл, изменяющий номер столбца от 1 до 10. Для вычисления произведения текущей строки матрицы x на текущий столбец матрицы y**,** в теле внутреннего цикла требуется организовать еще один цикл, параметр которого меняется от 1 до 20.

### 2.4.8. Контрольные вопросы по теме «Базовые алгоритмы и примеры их реализации»

1. Что называется алгоритмом?
2. Какими основными свойствами должен обладать алгоритм?
3. Какие существуют способы описания алгоритмов?
4. Какими графическими символами принято изображать в схемах алгоритма:

* операции ввода/вывода данных;
* операции над данными;
* выбор направления выполнения алгоритма в зависимости от результата выполнения условия;
* организацию регулярного цикла;
* начало и конец обработки данных;
* комментарии к элементам схемы алгоритма?

1. В чем отличие цикла структуры с предусловием от цикла структуры с постусловием?
2. Что такое параметр цикла?
3. В чем отличие регулярной циклической структуры от итеративной циклической структуры?

### 2.4.9. Тестовые задания по теме «Базовые алгоритмы и примеры их реализации»

1. **Для ввода данных в схемах алгоритма используют фигуру**
2. параллелограмм
3. ромб
4. прямоугольник
5. треугольник
6. **Укажите правильный результат работы встроенной функции INT(x)**

**при x=-275**

1. -3
2. -2
3. -275
4. -25
5. **Укажите правильный результат выполнения программной строки INT(x - FIX(x)) при x= -825**
6. -1
7. 0
8. 8
9. -16
10. **Значение переменной Y в результате выполнения следующих операторов**

**Y = 5**

**X = 2**

**Y = (Y – 2 \* X) / 2 + X** будет равно

1. 2,5
2. 0,5
3. 2
4. 5
5. **Операторы в программе, записанные на одной строке, разделяются**
6. двоеточием
7. запятой
8. точкой с запятой
9. нет верного ответа
10. **Укажите правильный результат выполнения следующего фрагмента программы при s=68**

**…**

**PRINT”сумма членов”**

**PRINT”арифметической прогрессии =”**

**PRINT s**

**…**

1. сумма членов арифметической прогрессии=68
2. сумма членов арифметической прогрессии=68
3. “сумма членов арифметической прогрессии”=68
4. “сумма членов” “арифметической прогрессии”=68
5. **Укажите правильный результат выполнения следующего фрагмента программы при x=3, y=15**

**…**

**PRINT FIX(x\*y+7) SPC (10) INT(x\*y-7)**

**…**

1. 11 -3
2. 12 -2
3. 11 -2
4. 12 -3
5. **Укажите правильный результат выполнения следующего фрагмента программы**

**…**

**DEFINT a-b**

**DEFSTR p-q**

**READ a, p, b, q**

**DATA 3, площадь, 4, прямоугольника**

**PRINT p SPC (1) q “=” a\*b**

**…**

1. площадь прямоугольника=12
2. площадь прямоугольника”=”3\*4
3. сообщение об ошибке
4. 3 площадь прямоугольника = 4
5. **Укажите правильный результат выполнения следующего фрагмента программы**

**…**

**DEFINT a-d**

**DEFSNG f**

**a=4... c=9... f=14**

**b=INT(c mod a-f)**

**d=FIX(c mod a –f)**

**PRINT b, d**

**…**

1. -1 0
2. -1 -1
3. 0 0
4. 0 -1
5. **Определите результат работы программы**

**A = -5 ... В = -6 ... А = А + 2 \* В ... В = А ... PRINT A B …**

1. –17 -17
2. 0 -17
3. –6 -6
4. нет верного ответа
5. **Логически неверно записанной является последовательность команд**
6. PRINT S ... А = A \* 2 ... INPUT A, В ... S = A + B \*
7. А = A \* 2 ... INPUT A, В ... S = A + B ... PRINT S
8. INPUT A, В ... S = A + B ... А = A \* 2 ... PRINT S
9. INPUT A, В ... А = A \* 2 ... S = A + B ... PRINT S
10. **Укажите правильный результат выполнения следующего фрагмента программы**

…

**READ x, y, a, b, c**

**PRINT a “=” x\*y, b “=” c**

**DATA 125, 7, площадь, результат, 128**

**…**

1. сообщение об ошибке
2. площадь=875 результат=128
3. площадь=125\*7 результат=128
4. площадь”=”875 результат”=”128
5. Укажите правильный результат выполнения следующего фрагмента программы

**…**

**READ a, b, c, x, y**

**PRINT “a=”a “b=”b “c=”c x\*y**

**DATA -45, 172, 3, 4**

**…**

1. сообщение об ошибке
2. a=-45b=172c=34
3. “a=” -44 ”b=”172”c=”3 4
4. a=-45 b=173 c=3 12
5. **Укажите правильный результат выполнения следующего фрагмента программы**

**…**

**a= 35... b=4... c=12**

**d=a\*b-c**

**DEFINT b, d**

**DEFSNG a, c**

**PRINT “d=” d**

**…**

1. d=0 \*
2. d=128
3. d=13
4. d=12
5. **Что будет на экране в результате работы фрагмента программы**

…

**DEFINT n , m**

**DEFSNG a, t, b**

**a=24 ... b=29**

**n=a ... m=b ... t=a + n**

**PRINT n, m, t**

**….**

1. 2 3 44
2. 2 2 44
3. 24 29 48
4. 3 3 54
5. **Что будет на экране в результате работы фрагмента программы**

…

**DEFINT n, m, k, l**

**DEFSNG a, b**

**a=26 ... b= -26**

**n=INT(a) ... m=FIX(a)**

**k=FIX(b) ... l=INT(b)**

**PRINT n, m, k, l**

**…**

1. 2 2 -2 -3
2. 3 2 -3 -3
3. 2 2 -2 -2
4. 3 2 -2 -3
5. **Что будет на экране в результате работы фрагмента программы**

…

**DEFINT n, m, c, k, l, i**

**DEFSNG a, b**

**READ a, b**

**DATA 28, 33**

**c=FIX(a+b)\*7**

**k=c MOD 5 ... l=c\5**

**i=INT(b) MOD 5**

**PRINT k, l, i**

**…**

1. 2 8 3
2. 2 8 0
3. 04 8 06
4. 8 2 3
5. **Что будет на экране в результате работы фрагмента программы**

…

**PRINT “Алгоритмический”**

**PRINT “ язык”**

**PRINT “ BASIC”**

**…**

1. Алгоритмический язык BASIC
2. Алгоритмический

язык

BASIC

1. Алгоритмический языкBASIC
2. “Алгоритмический ””язык””BASIC”
3. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**X=2... Y=3**

**IFX>YTHENS=X\*YELSES=X+Y**

**PRINTS**

1. 5
2. 6
3. 2
4. 3
5. **Определить, какое значение переменной** Y **будет присвоено в результате выполнения фрагмента программы**

**…**

**X= -1**

**IF ABS(X+1)>1 THEN Y=1 ELSE Y=ABS(X+1)**

**…**

1. 0
2. -1
3. 2
4. 1
5. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**…**

**A=2... B=3**

**IF A<B THEN S=A\*B ELSE S=A+B**

**PRINT S**

**…**

1. 6
2. 1
3. 3
4. 2
5. **Определить, какое значение переменной** Y **будет присвоено в результате выполнения фрагмента программы**

**…**

**X= -1**

**IF ABS(X) <=1 THEN Y= ABS(X) ELSE Y=SQR(1-X\*X)**

**…**

1. 1
2. 0
3. 2
4. -1
5. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**…**

**C=3... D=3**

**IF C<D THEN S=C\*D ELSE S=C+D**

**PRINT S**

**…**

1. 6
2. 9
3. 5
4. 3
5. **Разветвляющийся алгоритм может быть**
6. стандартным, вложенным, усеченным
7. стандартным, вложенным, усеченным, регулярным
8. стандартным, вложенным, усеченным, итерационным
9. нет верного ответа
10. **Разветвляющиеся алгоритмические структуры программируются операторами**
11. условного перехода IF, выбора SELECTCASE
12. условного перехода IF, выбора SELECTCASE, переключателя DO LOOP
13. условного перехода IF
14. нет верного ответа
15. **Что будет напечатано в результате выполнения программы**

**…**

**READ R**

**IF R = 0 THEN Y=R\*10**

**IF R > 0 THEN Y=R+10**

**IF R < 0 THEN Y=R\*R/2**

**PRINT Y**

**DATA –10**

**…**

1. 50
2. 0
3. 100
4. –10
5. **Что будет напечатано в результате выполнения программы**

**…**

**DEFSNG x - z**

**READ x, y**

**DATA 66, 58**

**IF x>y THEN z=x+y ... x=y**

**IF x<=y THEN z=x-y**

**PRINT x, y, z**

**…**

1. 58 58 124
2. 66 58 08
3. 58 58 0
4. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT a – c, z**

**READ a, b, c**

**DATA 15, 10, 25**

**IF a>b AND a<c THEN**

**z=a**

**ELSEIF b>c THEN**

**z=b**

**ELSE**

**z=c**

**END IF**

**PRINT “z=” z**

**…**

1. z=15
2. z=10
3. z=25
4. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT k - m**

**READ k, m, l**

**DATA 5, 15, 10**

**IF k<m AND m<l THEN**

**p=k**

**ELSE**

**IF k>m OR m>l THEN**

**p=m**

**ELSE**

**p=l**

**END IF**

**END IF**

**PRINT “p=”p**

**…**

1. p=5
2. p=10
3. p=15
4. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT a - d, x**

**READ a, b, c, d, x**

**DATA 25, 30, 5, 10, 15**

**IF a>b AND b>c AND c<d AND d>x THEN**

**x=a**

**ELSEIF a<b OR b>c AND c<d OR d>x THEN**

**x=b**

**ELSEIF a>b OR b>c OR c<d OR d>x THEN**

**x=c**

**ELSE**

**x=d**

**END IF**

**PRINT “x=” x**

**…**

1. x=25
2. x=30
3. x=5
4. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT k - l**

**READ k, l**

**DATA 5, 7**

**IF k<>l THEN**

**IF k>l THEN**

**l=k**

**ELSE**

**k=l**

**END IF**

**ELSE**

**k=0 ... l=0**

**END IF**

**PRINT “k=” k, “l=” l**

**…**

1. k=5 l=5
2. k=7 l =7
3. k=0 l=0
4. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = 0 ... K = 0**

**IF N THEN K=K+10**

**PRINT K**

**…**

1. 0
2. 10
3. 100
4. сообщение об ошибке
5. нет верного ответа
6. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = 0 ... K = 0**

**IF N THENK=K+1 ELSE K = K- 1**

**PRINT K**

**…**

1. -1
2. 0
3. 1
4. сообщение об ошибке
5. нет верного ответа
6. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = 2 ... K = 0**

**IF N THEN K=K+10 ELSE K=K-1**

**PRINT K**

**…**

1. 10
2. 0
3. -1
4. сообщение об ошибке
5. нет верного ответа
6. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = -2 ... K = 0**

**IF N THEN K=K+1 ELSE K=K-1**

**PRINT K**

**…**

1. 1
2. 0
3. -1
4. сообщение об ошибке
5. нет верного ответа
6. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = -2 ... K = 0**

**SELECT CASE N**

**CASE 0**

**K=K+1**

**CASE ELSE**

**K=K-1**

**END SELECT**

**PRINT K**

…

1. -1
2. 0
3. 1
4. -2
5. сообщение об ошибке
6. нет верного ответа
7. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = -2 ... K = 0**

**SELECT CASE N**

**CASE IS<0**

**K=K+1**

**CASE ELSE**

**K=K-1**

**END SELECT**

**PRINT K**

**…**

1. 1
2. 0
3. -1
4. -2
5. сообщение об ошибке
6. нет верного ответа
7. **Что будет напечатано в результате выполнения программы**

**…**

**DEFINT K , N**

**N = -2 ... K = 0**

**SELECT CASE N**

**CASE 0**

**K=K+10**

**CASE ELSE**

**K=K-10**

**PRINT K**

**…**

1. сообщение об ошибке 0
2. 10
3. -2
4. –10
5. 0
6. нет верного ответа
7. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**A=3... B=4**

**IF A>B THEN**

**A=A+1**

**S=A\*B**

**ELSE**

**B=B+1**

**S=A+B**

**END IF**

**PRINT S**

1. 8
2. 12
3. 4
4. 3
5. **Определить, какое значение переменной S будет напечатано в результате выполнения фрагмента программы**

…

**X=3... Y=4**

**IF X<3 THEN Y=Y+1 ELSE X=X+1**

**S=X\*Y**

**PRINT S**

**…**

1. 16
2. 12
3. 15
4. 9
5. **Оператор** IF **сначала**
6. проверяет свое условное выражение
7. выполняет операторы из тела программы, идущие после него
8. подключает процедуры
9. передает управление на конец программы
10. **Определите условие того, что месяц является зимним...**
11. MES=12 OR MES=1 OR MES=2
12. MES =12 AND MES =1 AND MES =2
13. MES =12 AND MES <=3
14. MES =12 AND (MES =1 OR MES=2)
15. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**…**

**A=3... B=3**

**IF A<B THEN**

**S=A\*B**

**S=S+1**

**ELSE**

**S=A+B**

**S=S-1**

**END IF**

**PRINT S**

**…**

1. 5
2. 6
3. 8
4. 10
5. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**X=2... Y=3**

**IF X>Y THEN S=X\*Y ELSE S=X+Y**

**PRINT S**

1. 5
2. 6
3. 2
4. 3
5. **Зимние каникулы начинаются 29 декабря и заканчиваются 10 января. Выберите правильный ответ ... условный оператор должен вывести сообщение ”Каникулы”**
6. IF (MES=12 AND DAY>=29)OR (MES=1 AND DAY<=10)THEN

PRINT”Каникулы”

1. IF (MES=12 OR DAY>=29) AND (MES=1 OR DAY<=10) THEN

PRINT”Каникулы”

1. IF (MES=12 AND DAY>=29) OR (MES=1 OR DAY<=10) THEN

PRINT”Каникулы”

1. IF (MES=12 OR DAY>=29) OR (MES=1 AND DAY<=10) THEN

PRINT”Каникулы”

1. **Определить, какое значение переменной** Y **будет присвоено в результате выполнения фрагмента программы**

**…**

**X= -1**

**IF ABS(X+1)>1 THEN Y=1 ELSE Y=ABS(X+1)**

**…**

1. 0
2. -1
3. 2
4. 1
5. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**…**

**A=2... B=3**

**IF A<B THEN S=A\*B ELSE S=A+B**

**PRINT S**

**…**

1. 6
2. 1
3. 3
4. 2
5. **Определить, какое значение переменной** Y **будет присвоено в результате выполнения фрагмента программы**

**…**

**X= -1**

**IF ABS(X) <=1 THEN Y= ABS(X) ELSE Y=SQR(1-X\*X)**

**…**

1. 1
2. 0
3. 2
4. -1
5. **Определить, какое значение переменной** S **будет напечатано в результате выполнения фрагмента программы**

**…**

**C=3... D=3**

**IF C<D THEN S=C\*D ELSE S=C+D**

**PRINT S**

**…**

1. 6
2. 9
3. 5
4. 3
5. **Значение переменной А после выполнения фрагмента программы**

**…**

**A=0**

**FOR B=1 TO 10**

**A=A+2**

**NEXT B**

**…**

**будет равно**

1. 20
2. 21
3. 22
4. 24
5. **Значение переменной M после выполнения фрагмента программы**

**…**

**M=0**

**FOR N=1 TO 3**

**FOR K=1 TO 2**

**M=M+1**

**NEXT K**

**NEXT N**

**…**

**будет равно**

* 1. 6
  2. 13
  3. 12
  4. нет верного ответа

1. **Значение переменной** M **после выполнения фрагмента программы**

**…**

**M=1**

**FOR N=1 TO 4**

**FOR K=1 TO 3**

**M=M+1**

**NEXT K , N**

**…**

**будет равно**

1. 13
2. 21
3. 12
4. будет сообщение об ошибке
5. **Значение переменной** M **после выполнения фрагмента программы**

**…**

**M=2**

**FOR N=1 TO 5**

**FOR K=1 TO 2**

**M=M+1**

**NEXT N**

**NEXT N**

**…**

**будет равно**

1. будет сообщение об ошибке\*
2. 12
3. 11
4. нет верного ответа
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT A , I**

**A=0**

**FOR I = 1 TO –1**

**A=A+1**

**NEXT I**

**PRINT A**

**…**

1. 0
2. 3
3. 2
4. нет верного ответа
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT A , I**

**A=2**

**FOR I = -1 TO 1 STEP -1**

**A=A -1**

**NEXT I**

**PRINT A**

**…**

1. 2
2. -1
3. 0
4. нет верного ответа
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT A , I**

**A=0**

**FOR I = -1 TO –1**

**A=A+1**

**NEXT I**

**PRINT A**

**…**

1. 1
2. 3
3. 0
4. нет верного ответа
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT A , I**

**A=0**

**FOR I=1 TO 31**

**A=A+1**

**NEXT I**

**PRINT A**

**…**

1. 31
2. 30
3. 0
4. нет верного ответа
5. **Для нахождения наименьшего значения числовой функции от аргумента , значение которого изменяется на заданном интервале, за начальное наименьшее значение следует принять**
6. число, близкое к наибольшему числу из диапазона типа данных рассматриваемой функции
7. число, близкое к наименьшему числу из диапазона типа данных рассматриваемой функции
8. 0
9. нет верного ответа
10. **Для нахождения наибольшего значения числовой функции от аргумента , значение которого изменяется на заданном интервале, за начальное наибольшее значение следует принять**
11. число, близкое к наименьшему числу из диапазона типа данных рассматриваемой функции
12. число, близкое к наибольшему числу из диапазона типа данных рассматриваемой функции
13. 0
14. нет верного ответа
15. **Для нахождения значения произведения числовой функции от аргумента , значение которого изменяется на заданном интервале, за начальное значение произведения следует принять**
16. 1
17. число, близкое к наибольшему числу из диапазона типа данных рассматриваемой функции
18. число, близкое к наименьшему числу из диапазона типа данных рассматриваемой функции
19. 0
20. нет верного ответа
21. **Для нахождения значения суммы числовой функции от аргумента , значение которого изменяется на заданном интервале, за начальное значение суммы следует принять**
22. 0
23. число, близкое к наибольшему числу из диапазона типа данных рассматриваемой функции
24. число, близкое к наименьшему числу из диапазона типа данных рассматриваемой функции
25. 1
26. нет верного ответа
27. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT A , I**

**A=0**

**FOR I=1 TO 31 STEP -3**

**A=A+1**

**NEXT I**

**PRINT A**

**…**

1. 0
2. 30
3. 31
4. нет верного ответа
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT A, I - J**

**A=0**

**FOR I=1 TO 3**

**A=A+1**

**FOR J = 1 TO -1**

**A=A+1**

**NEXT J**

**NEXT I**

**PRINT A**

**…**

1. 3
2. 9
3. 0
4. нет верного ответа
5. будет сообщение об ошибке
6. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT i, k, h**

**k=2 ... h=2**

**FOR i=0 TO -2 STEP -1**

**k=k + h**

**NEXT i**

**PRINT k**

**…**

1. 8
2. 2
3. 6
4. -4
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT i, s**

**FOR i=1 TO 5**

**s =s+3**

**NEXT i**

**PRINT “i=”i, ”s=”s**

**…**

1. i=6 s=15
2. i=5 s=15
3. i=5 s=12
4. i=6 s=18
5. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT i, s**

**s=0**

**FOR i=2 TO 50 STEP 5**

**s=s+i**

**IF i=27 THEN EXIT FOR**

**NEXT i**

**PRINT s**

**…**

1. 87
2. 50
3. 103
4. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT i, s**

**s=15**

**FOR i=5 TO 15 STEP -2**

**s=s+1**

**NEXT i**

**PRINT s**

**…**

1. 15
2. 25
3. 19
4. 20
5. 18
6. **Что будет результатом выполнения фрагмента программы**

**…**

**DEFINT k,s**

**s=0**

**FOR k=2 TO -2 STEP -1**

**s=s+k ... k=k-2**

**NEXT k**

**PRINT “s=”s, “k=”k**

**…**

1. s=-1 k=-4
2. s=0 k=-3
3. s=1 k=-2
4. s=3 k=-1
5. **C использованием итеративных циклических структур можно вычислять**
6. все утверждения верны
7. значения функции в заданном диапазоне, произведения и суммы с конечным числом слагаемых и сомножителей
8. произведения и суммы конечных или бесконечных рядов
9. значения членов конечных и бесконечных рядов
10. все утверждения неверны
11. **Преобразование итеративного цикла с предусловием в цикл с постусловием**
12. всегда возможно
13. возможно иногда
14. невозможно
15. **Преобразование итеративного цикла с постусловием в цикл с предусловием**
16. всегда возможно
17. возможно иногда
18. невозможно
19. **Результатом работы фрагмента программы**

**…**

**DEFINT N**

**N=0**

**DO WHILE N<5**

**N=N+1**

**LOOP**

**PRINT N**

**…**

**будет**

1. вывод на экран 5
2. вывод на экран 6
3. вывод на экран 0
4. сообщения об ошибке
5. нет верного ответа
6. «зацикливание»
7. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFINT N**

**N=0**

**DO UNTIL N<5**

**N=N+1**

**LOOP**

**PRINT N**

**…**

1. 0
2. 6
3. 5
4. сообщение об ошибке
5. нет верного ответа
6. «зацикливание»
7. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFINT N**

**N=0**

**DO**

**PRINT “#”**

**N=N+1**

**LOOP UNTIL N**

**…**

1) #

1. ##
2. ничего не будет
3. сообщение об ошибке
4. нет верного ответа
5. «зацикливание»
6. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFINT N**

**N=0**

**DO WHILE N**

**PRINT “#”**

**N=N+1**

**LOOP**

**…**

1. ничего не будет
2. #
3. ##
4. сообщение об ошибке
5. нет верного ответа
6. «зацикливание»
7. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFINT N**

**N=0**

**DO**

**PRINT “\*”**

**N=N+1**

**LOOP WHILE N**

**…**

1. «зацикливание»#
2. ничего не будет
3. \*
4. \*\*
5. сообщение об ошибке
6. нет верного ответа
7. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFSNG s**

**DEFINT i**

**s=0 : i=0**

**DO UNTIL i<5**

**i=i+1 : s=s+1/i**

**LOOP**

**PRINT “s=”; s**

**…**

1. s=0#
2. s=0.2
3. s=2.283333
4. s=2.083333
5. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFSNG s**

**DEFINT k**

**s=1 : k=1**

**DO WHILE k>1**

**s=s+1/k : k=k-1**

**LOOP**

**PRINT “s=”; s**

**…**

1. s=1\*
2. s=2
3. Сообщение об ошибке: “Деление на ноль”
4. Ничего, произойдет “зацикливание”
5. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFSNG s**

**DEFINT i**

**s=0 : i=1**

**DO**

**s=s+1/i : i=i-1**

**LOOP UNTIL i<=1**

**PRINT “s=”; s**

**…**

1. s=1#
2. s=0
3. Сообщение об ошибке: “Деление на ноль”
4. Ничего, произойдет “зацикливание”
5. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFINT n, s**

**s=0 : n=123**

**DO**

**s=s\*10 + n MOD 10**

**n=n\10**

**LOOP UNTIL n=0**

**PRINT “s=”;s,“n=”;n**

**…**

1. s=321 n=0#
2. s=123 n=0
3. s=321 n=123
4. s=0 n=0
5. **Что будет на экране в результате работы фрагмента программы**

**…**

**DEFINT n, s, a**

**s=0 : n=346**

**DO**

**a= n MOD 10**

**s=s+a**

**n=n\10**

**LOOP UNTIL n=0**

**PRINT “s=”;s,“a=”;a**

**…**

1. s=13 a=3#
2. s=10 a=3
3. s=13 a=6
4. s=7 a=4
5. **Метод пузырька – это** 
   1. сортировка методом прямого обмена\*
   2. сортировка методом выбора
   3. сортировка методом простого перебора
   4. сортировка методом бинарных вставок
6. **Поиск заданного элемента производится в** 
   1. в произвольном массиве
   2. в упорядоченном массиве
   3. в неупорядоченном массиве
   4. в списке нет правильного ответа
7. **Выбор максимального и минимального элемента целесообразно проводить** 
   1. в упорядоченном массиве
   2. в произвольном массиве
   3. в неупорядоченном массиве
   4. в сжатом массиве
8. **В упорядочивании элементов массива по убыванию методом выбора каждый раз происходит выбор** 
   1. максимального элемента
   2. минимального элемента
   3. соседнего элемента
   4. предыдущего элемента
9. **Определите правильный результат выполнения следующего фрагмента программы**

**...**

**DIM A (1 TO 5) AS SINGLE**

**FOR I=1 TO 5**

**A (I) =I**

**NEXT I**

**PRINT A (I)**

**…**

1. сообщение об ошибке\*
2. 5
3. 6
4. 0
5. **После выполнения следующего фрагмента программы**

**…**

**FOR I=1 TO 7**

**PRINT A (I)**

**NEXT I**

**…**

**массивбудетнапечатан**

1. в столбец\*
2. в строку
3. в столбец со сдвигом
4. по диагонали экрана

33 **Следующий фрагмент программы**

**…**

**FORI=KTON-1**

**A (I) = A (I+1)**

**NEXT I**

**…**

**выполняет**

1. сжатие массива “ сдвигом влево”
2. сжатие массива “ сдвигом вправо”
3. удаление первых K элементов массива
4. удаление последних N-K элементов массива
5. **Следующий фрагмент программы**

**…**

**FOR I=1 TO N-1**

**IF A (I) =0 THEN**

**FOR J=I TO N-1**

**A (J) =A (J+1)**

**NEXT J**

**N=N-1**

**END IF**

**NEXT I**

**…**

1. удаляет из массива нулевые элементы
2. перемещает нулевые элементы влево
3. перемещает нулевые элементы вправо
4. оставляет все по-прежнему

**Что будет на экране после выполнения следующего фрагмента программы**

**…**

**FOR i=1 TO 5**

**a(i)=3-2\*i**

**NEXT i**

**FOR i=5 TO 1 STEP -1**

**PRINT a(i) " "**

**NEXT i**

**…**

1. 1 2 3 4 5
2. -1 -3 -5 -7
3. 5 4 3 2 1
4. -7 -5 -3 -1 1
5. **Что подсчитывает следующий фрагмент программы**

**…**

**k=0**

**s=0**

**FOR i=1 TO 5**

**IF a(i)=0 THEN k=k+1**

**IF a(i)<0 THEN s=s+a(i)**

**NEXT i**

**…**

1. количество элементов, равных 0, и сумму отрицательных элементов
2. сумму элементов, меньших 0, и количество положительных элементов
3. количество всех элементов и сумму положительных элементов

произведение отрицательных элементов и сумму элементов, равных 0

1. **Что выполняет следующий фрагмент программы**

**…**

**FOR i=1 TO c STEP 1**

**INPUT a(i)**

**NEXT i**

**j=0**

**FOR i=1 TO c STEP 1**

**IF a(i)>0 THEN**

**j=j+1... b(j)=a(i)**

**END IF**

**NEXT i**

**…**

1. формирует массив из положительных элементов массива a
2. упорядочивает массив по возрастанию
3. меняет максимальный и минимальный элементы местами
4. упорядочивает массив по убыванию
5. **Что выполняет следующий фрагмент программы**

**…**

**FOR i=1 TO c STEP 1**

**INPUT a(i)**

**NEXT i**

**m=a(1) ...im=1 ...n=a(1) ...in=1**

**FOR i=2 TO c STEP 1**

**IF a(i)>m THEN**

**m=a(i) ... im=i**

**END IF**

**IF a(i)<n THEN**

**n=a(i) ... in=i**

**END IF**

**NEXT i**

**a(in)=m ... a(im)=n**

**…**

1. меняет максимальный и минимальный элементы местами
2. упорядочивает массив по возрастанию
3. упорядочивает массив по убыванию
4. формирует массив из положительных элементов массива a
5. **Что выполняет следующий фрагмент программы**

**…**

**FOR i=1 TO n STEP 1**

**INPUT a(i)**

**NEXT i**

**FOR i=1 TO n-1**

**m=a(i) ... km=i**

**FOR j=i+1 TO n**

**IF a(j)<m THEN**

**m=a(j) ... km=j**

**END IF**

**NEXT j**

**a(km)=a(i) ... a(i)=m**

**NEXT i**

**…**

1. упорядочивает массив по убыванию
2. упорядочивает массив по возрастанию
3. меняет максимальный и минимальный элементы местами
4. формирует массив из положительных элементов массива
5. **Что выполняет следующий фрагмент программы**

**…**

**FOR i=1 TO c STEP 1**

**INPUT a(i)**

**NEXT i**

**FOR i=1 TO n-1 STEP 1**

**FOR j=i+1 TO n**

**IF a(i)<a(j) THEN**

**SWAP a(i), a(j)**

**END IF**

**NEXT j**

**NEXT i**

**…**

1. упорядочивает массив по убыванию\*
2. упорядочивает массив по возрастанию
3. меняет максимальный и минимальный элементы местами
4. формирует массив из положительных элементов массива a
5. **Укажите количество объявленных массивов**

**DIM A (0 to 3, 4)... DIM A (3, 4)... DIMA (3, 0 to 4)... DIMA (0 to 3, 0 to 4)**

* 1. 1
  2. 2
  3. 3
  4. 4

1. **Определите, какое значение приобретет элемент массива** A(3,3) **при выполнении следующего фрагмента программы**

**…**

**FOR I=1 TO 5**

**F0R J=1 TO 5**

**IF I<J THEN A (I, J) = I+J ELSE A (I, J) =I\*J**

**NEXT J**

**NEXT I**

**…**

1. 9
2. 6
3. 12
4. 3
5. **Определите, как будет напечатан двумерный массив после выполнения следующего фрагмента программы**

**…**

**FOR I=1 TO 7**

**FOR J=1 TO 5**

**PRINT A (I, J)**

**NEXT J**

**NEXT I**

**…**

1. в строку
2. в виде таблицы
3. в столбец
4. будет выдано сообщение об ошибке
5. **Определите, какое значение приобретет элемент массива A(2,3) при выполнении следующего фрагмента программы**

**…**

**FOR I=1 TO 5**

**F0R J=1 TO 5**

**IF I<J THEN A (I, J) = I+J ELSE A (I, J) =I\*J**

**NEXT J**

**NEXT I**

**…**

1. 5\*
2. 6
3. 2
4. 3
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**…**

**DIM A (1 TO 3, 1 TO 4) AS SINGLE**

**FOR I=1 TO 3**

**FOR J=1 TO 4**

**A (I, J) = I\*J**

**NEXT J**

**NEXT I**

**PRINT A (I, J)**

**…**

1. сообщение об ошибке
2. 12
3. 20
4. 15
5. **Определите, какое значение приобретет элемент массива C(4,3) при выполнении следующего фрагмента программы**

**…**

**FOR K=1 TO 5**

**F0R J=1 TO 5**

**IF K>J THEN C (K, J) = K+J ELSE C (K, J) =K\*J**

**NEXT J**

**NEXT K**

**…**

1. 7
2. 12
3. 4
4. 3
5. **Определите, как будет напечатан двумерный массив после выполнения следующего фрагмента программы**

**…**

**FOR I=1 TO 7**

**FOR J=1 TO 5**

**PRINT A (I, J)**

**NEXT I**

**PRINT**

**NEXT J**

**…**

1. будет выдано сообщение об ошибке
2. в виде таблицы
3. в строку
4. в столбец
5. **Определите, какое значение приобретет элемент массива C(3,3) при выполнении следующего фрагмента программы**

**…**

**FOR K=1 TO 5**

**F0R J=1 TO 5**

**IF K>J THEN C (K, J) = K+J ELSE C (K, J) =K\*J**

**NEXT J**

**NEXT K**

**…**

1. 9
2. 6
3. 12
4. 5
5. **Определите, какой тип распределения памяти имеет объявленный массив.**

**DIM A (1 TO 3, 1 TO 5) AS SINGLE**

1. статический\*
2. динамический
3. фактический
4. вещественный
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**...**

**FOR i=1 TO 3**

**FOR j=1 TO 3**

**INPUT a(i , j)**

**NEXT j**

**NEXT i**

**…**

1. вводит матрицу по строкам
2. вводит одномерный массив
3. вводит матрицу по столбцам
4. выводит на экран матрицу
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**…**

**FOR i=1 TO 4**

**r=a(i, 2)**

**a(i,2)=a(i,4)**

**a(i,4)=r**

**NEXT i**

**…**

1. меняет 2-ой и 4-ый столбцы матрицы
2. меняет элементы главной и побочной диагоналей
3. делает одинаковыми 2-ой и 4-ый столбцы
4. присваивает всем элементам матрицы значение переменной r
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**…**

**s=0**

**FOR i=1 TO 5**

**FOR j=1 TO i-1**

**s=s+a(i, j)**

**NEXT j**

**NEXT i**

**PRINT “s=” s**

**…**

1. выводит сумму элементов ниже главной диагонали
2. вводит сумму элементов выше главной диагонали
3. выводит сумму элементов главной диагонали
4. выводит сумму элементов побочной диагонали
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**…**

**P=1**

**FOR i=1 TO 3**

**FOR j=1 TO 3**

**IF a(i, j)>0 THEN**

**p=p\*a(i, j)**

**END IF**

**NEXT j**

**NEXT i**

**…**

1. вычисляет произведение положительных элементов
2. находит произведение всех элементов
3. вычисляет произведение положительных элементов в каждой строке
4. находит среднее арифметическое элементов матрицы
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**…**

**s=0**

**FOR i=1 TO 3**

**FOR j=1 TO 3**

**s=s+a(i, j)^2**

**NEXT j**

**NEXT i**

**s=s/9**

**…**

1. находит среднее арифметическое квадратов элементов матрицы
2. находит среднее арифметическое элементов матрицы
3. находит сумму квадратов элементов в каждом столбце
4. находит сумму всех элементов
5. **Определите, что будет выведено на экран после выполнения следующего фрагмента программы**

**…**

**FOR i=1 TO 5**

**FOR j=1 TO 5**

**IF j mod 2=0 THEN**

**a(i,j)=0**

**END IF**

**NEXT j**

**NEXT i**

**…**

1. заполняет нулями элементы четных столбцов
2. заполняет четные строки нулями
3. заполняет нулями четные элементы матрицы
4. выводит сумму элементов побочной диагонали
5. **Определить результат выполнения фрагмента программы**

**А = LEN («Маша ела кашу») ... PRINT A**

1. 13
2. 11
3. «Маша ела кашу»
4. Маша ела кашу
5. **Определить результат выполнения фрагмента программы**

**А$ = MID$ («стереопроигрыватель», 1, 6) ... PRINTA$**

1. стерео
2. игра
3. нет верного ответа
4. проигрыватель
5. **С помощью программы**

**…**

**INPUT “введите слово” A$**

**FOR I = LEN(A$) TO 1 STEP –1**

**PRINT MID$(A$, I ,1)**

**NEXT I**

**…**

**выполняются следующие действия**

1. печатаются вырезаемые символы введенного слова в обратном порядке
2. печатается введенное слово
3. печатаются вырезаемые символы введенного слова по одному
4. печатаются вырезанные символы введенного слова в обратном порядке через один
5. нет верного ответа
6. **В результате преобразований, описанных программной строкой**

**A$ = MID$ («математика», 1, 2) + MID$ («математика», 5, 2)**

**будут выполнены следующие действия**

1. в переменную A$ будут записаны первые две буквы и пятая и шестая буквы выбранного слова
2. в переменную A$ будут записаны первые две буквы выбранного слова
3. в переменную A$ будут записаны пять букв выбранного слова
4. в переменную A$ будут записаны первые две буквы и последние пять букв выбранного слова
5. **В результате выполнения фрагмента программы**

**…**

**A$=”КОЗАК”**

**MID$(A$ , 2, 1) = “A”**

**PRINTA$**

**…**

1. вторая буква будет заменена на букву «А»
2. все буквы будут заменены на букву «А»
3. будет вырезана буква «А»
4. будет вырезана вторая буква слова
5. **Результат выполнения фрагмента программы**

**…**

**A$ = «электростанция»**

**PRINT MID$ (A$, 2, 3) + MID$ (A$, 12, 3)**

**...**

1. лекция
2. станция
3. рост
4. 6
5. **Результат выполнения фрагмента программы**

**…**

**C$ = «лефортово»**

**В = LEN (C$) + LEN (C$)**

**PRINT В**

**…**

**будет**

1. 18
2. 9 + 9
3. лефортоволефортово
4. 2 лефортово
5. **Из слова ИНФОРМАТИКА получено слово ФОРМАТ. Вставьте пропущенные значения в функцию MID$ («ИНФОРМАТИКА», … )**
6. 3, 6
7. 6, 3
8. 3, 8
9. 8, 3
10. **Результат выполнения фрагмента программы**

**PRINT ”24+24 =” 24 +24 …**

1. 24+24 =48
2. 24+24 = 24+24
3. 48 = 48
4. 96
5. нет верного ответа
6. **Результат выполнения фрагмента программы**

**….**

**A$=”Маша”**

**MID$(A$, 3, 2)=”ма”**

**PRINT A$**

**…**

1. Мама
2. Мамаша
3. Маша
4. сообщение об ошибке
5. **Результат выполнения фрагмента программы**

**…**

**DEFINT L , V**

**V1 =”123” ...V2 =”-123”**

**L1=LEN(STR$(V1))**

**L2=LEN(STR$(V2))**

**PRINT L1 L2**

**…**

1. 4 4
2. 3 3
3. 3 4
4. сообщение об ошибке
5. **Результат выполнения фрагмента программы**

**…**

**DEFSNG A,B**

**X$=” 12356” ...Y=”12356 “ ... Z$=”MAMA”**

**A=VAL(X$) ... B=VAL(Y$) ... C$=VAL(Z$)**

**PRINT A B C**

**…**

1. 12356 0 0
2. 12356 12356 0
3. 12356 12356 МАМА
4. сообщение об ошибке
5. **Чтобы получить слово ЖЕЛЕЗО из слова ЖЕЛЕЗНОДОРОЖНИК, имея**

**A$ = “ЖЕЛЕЗНОДОРОЖНИК ”нужно применить**

1. все ответы верны
2. MID$(A$, 1, 5) + MID$(A$, 7, 1)
3. MID$(A$, 1, 5) + MID$(A$, 9, 1)
4. LEFT$(A$, 5) + MID$(A$, 7, 1)
5. нет верного ответа

1. **В результате выполнения следующей программы**

**…**

### **DEFINT k**

**READ a$, k**

**DATA INFORMATION, 2**

**b$= MID$(a$, k+1, k+3)**

**PRINT b$**

**END**

**…**

**будет напечатано**

1. FORMA
2. FORM
3. INFORM
4. FORMATION
5. **В результате выполнения следующей программы**

**…**

**READ a$**

**DATA ABCDEFGH**

**b$= STRING$(5, “T”)**

**MID$(a$,5,6)=b$**

**PRINT a$**

**END**

**…**

**будет напечатано**

1. ABCDTTTT\*
2. ABCDTTTTTT
3. ABCDEFGHT
4. ABCDTTTTT
5. **В результате выполнения следующей программы**

**…**

**READ a$**

**DATA INFORM**

**b$= LEFT$(a$, 2)**

**c$=RIGHT$(a$,2)**

**t$=c$+MID$(a$, 3, 2) + b$**

**PRINT t$**

**END**

**…**

**будет напечатано**

1. FORMIN
2. INRMFO
3. NIFOMR
4. FORMA
5. **В результате выполнения следующей программы**

**…**

### **DEFINT f**

**a$=”CAT” ... b$=”CAP” ... c$=”CA”**

**f=0**

**IF a$<b$ THEN**

**f=1**

**ELSEIF b$<c$ THEN**

**f=2**

**ELSE**

**f=3**

**END IF**

**PRINT “f=” f**

**END**

**…**

**будет напечатано**

1. F=3
2. F=1
3. F=2
4. F=0
5. **В результате выполнения следующей программы**

**…**

**DEFINT i**

**READ a$**

**DATA MMKHT**

**b$= “”**

**FOR i=1 TO LEN(a$)**

**b$=b$+MID$(a$, LEN(a$) + 1 - i, 1)**

**NEXT i**

**PRINT b$**

**END**

**…**

**будет напечатано**

1. MKTHH\*
2. MKT
3. THKMM
4. TKM
5. **В результате выполнения следующей программы**

**INFORAMTION**

### **DEFINT k**

**READ a$, b$, k**

**DATA INFORMTION, A, 7**

**a$= LEFT$(a$, k-1)+b$+MID$(a$, k)**

**PRINT a$**

**END**

**…**

**будет напечатано**

1. **INFORMATION**
2. **INFORMAT**
3. **INFORM**
4. **В результате выполнения следующей программы**

**…**

**DEFINT i, j, n**

**READ n**

**DATA 4,символ,массив**

**DATA элемент,цикл**

**DIM a$(1 TO n)**

**FOR i=1 TO n**

**READ a$(i)**

**NEXT i**

**FOR i=1 TO n-1**

**FOR j=i+1 TO n**

**IF a$(i)>a$(j) THEN**

**b$ = a$(i)**

**a$(i)=a$(j)**

**a$(j)=b$**

**END IF**

**NEXT j, i**

**FOR i=1 TO n**

**PRINT a$(i)SPACE$(2)**

**NEXT i**

**END**

**…**

**будет напечатано**

1. массив символ цикл элемент
2. символ массив элемент цикл
3. цикл элемент массив символ
4. цикл элемент символ массив
5. **Что будет на экране в результате работы фрагмента программы**

**…**

### **DEFSNG a-b,x**

**INPUT a, b, x ‘a=2 b=3 x=1**

**PRINT “Значение ф-ции=” FNF(x)**

**DEFFNF (t)=a\*t^2+2\*t-5**

**END**

**…**

### сообщение об ошибке

### значение ф-ции= -5значение ф-ции= 5

### значение ф-ции= 0

1. **Что будет на экране в результате работы фрагмента программы**

**...**

### **EFINT n**

**n=123**

**PRINT “Summa=”**

**PRINT Sum%((n)), “chislo=”n**

**END**

**FUNCTION Sum%(x AS INTEGER)**

**DIM s AS INTEGER**

**DIM q AS INTEGER**

**s=0**

**DO**

**q=x MOD 10**

**s=s+q ... x=x\10**

**LOOP UNTIL x=0**

**Sum%=s**

**END FUNCTION**

**…**

### Summa = 6 chislo=123

### Summa = 3 chislo=123

### Summa =123 chislo=0

### Summa = 0 chislo=123

1. **Что будет на экране в результате работы фрагмента программы**

**...**

### **DEFINT i, k**

**k=4**

**DATA 1, 2, 3, 4**

**DIM a(1 TO k) AS SINGLE**

**FOR i=1 TO k**

**READ a(i)**

**NEXT i**

**PRINT “Ср Арифм =”Sum(a())/k**

**END**

**FUNCTION Sum (b() AS SINGLE)**

**DIM s AS SINGLE**

**s=0**

**FOR j=1 TO k**

**s=s+b(j)**

**NEXT j**

**Sum=s**

**END FUNCTION**

**…**

### Ср Арифм=0

### Ср Арифм=25

### Сообщение об ошибке

### Ср Арифм=1

1. **Что будет на экране в результате работы фрагмента программы**

**…**

### **DEFINT x-z, d**

**x=4 ... y=5 ... z=1**

**CALL DISCR(x, y, z)**

**PRINT “дискриминант=”d**

**END**

**SUB DISCR(a, b, c, d)**

**d=SQR(b^2-4\*a^c)**

**END SUB**

**…**

### сообщение об ошибке

### дискриминант = 3

### дискриминант = 4

### дискриминант = 25

1. **Что будет на экране в результате работы фрагмента программы**

**...**

FUNCTION INTEG(r AS SINGLE, p AS SINGLE, m AS INTEGER)

DEFSNG h, s, t

s=0 ... h=(p-r)/m

s=FNF(r)+FNF(p)

FOR t=r+h TO p-h STEP h

s=s+2\*FNF(t)

NEXT t

INTEG=s\*h/2

END FUNCTION

### **DEFSNG a-b**

DEFINT n

DEF FNF(t)=t^2

a=2 ... b=4

n=2

PRINT INTEG(a,n,b) “при n=”n

END

…

### **Сообщение об ошибке**

### **38 при n=2**

### **19 при n=2**

### **27 при n=2**

1. **Что будет на экране в результате работы фрагмента программы**

**...**

DEFINT i, n

DEFSNG s, p

s=0 ... n=3

DIM x(3) AS SINGLE

x(0)=4 ... x(1)=2 ... x(3)=6

CALL SP(x(), s, p)

PRINT “s=” s, “p=” p

END

SUB SP(x(), s, p)

s=0 ... p=1

FOR i=0 TO n

s=s+x(i) ... p=p\*x(i)

NEXT i

### **ND SUB**

…

### **s=4 p=4**

### **s=12 p=0**

### **s=12 p=36**

### **s=0 p=1**

1. **Что будет на экране в результате работы фрагмента программы**

**...**

### **DEFINT i, k**

K=4

DATA 1, 2, 3, 4

DIM a(1 TO k) AS SINGLE

FOR i=1 TO k

READ a(i)

NEXT i

PRINT “Ср Арифм=”Sum( (k))/k

END

FUNCTION Sum (n AS INTEGER)

SHARED a() AS INTEGER

DIM j AS SINGLE

s=0

FOR j=1 TO n

s=s+a(j)

NEXT j

Sum=s

END FUNCTION

…

### **Ср Арифм=25**

### **Ср Арифм=0**

### **сообщение об ошибке**

### **Ср Арифм=1**