МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВПО «Воронежский государственный технический университет»

Кафедра «Ракетные двигатели»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения курсовой работы по дисциплине
«Алгоритмические языки и программирование»
для студентов специальности 160700.65, 24.05.02
«Проектирование авиационных и
ракетных двигателей» очной формы обучения

Воронеж 2015

Составители: канд. физ.-мат. наук А.М. Сушков,
канд. техн. наук Д.П. Шматов

УДК 681.3

Методические указания для выполнения курсовой работы по дисциплине «Алгоритмические языки и программирование» для студентов специальности 160700.65, 24.05.02 «Проектирование авиационных и ракетных двигателей» очной формы обучения / Воронеж. гос. техн. ун-т; Сост. А.М. Сушков, Д.П. Шматов. Воронеж, 2015. 23 с.

В методических указаниях для выполнения курсовой работы содержатся набор заданий и пояснения для создания компьютерной программы на языке Паскаль.

Издание соответствует требованиям Федерального государственного образовательного стандарта высшего профессионального образования по направлению 160700.65, 24.05.02 «Проектирование авиационных и ракетных двигателей», дисциплине «Алгоритмические языки и программирование».

Рецензент д-р техн. наук, проф. Г.И. Скоморохов

Ответственный за выпуск зав. кафедрой д-р техн. наук, проф. В.С. Рачук.

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ВВЕДЕНИЕ

Основным принципом **модульного программирования** является принцип «разделяй и властвуй». **Модульное программирование** – это организация программы как совокупности небольших независимых блоков, называемых **модулями**, структура и поведение которых подчиняются определенным правилам.

ИСПОЛЬЗОВАНИЕ МОДУЛЬНОГО ПРОГРАММИРОВАНИЯ

Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Термин «модуль» в программировании начал использоваться в связи с внедрением модульных принципов при создании программ. В 70-х годах под модулем понимали какую-либо процедуру или функцию, написанную в соответствии с определенными правилами. Например: «Модуль должен быть простым, замкнутым (независимым), обозримым (от 50 до 100 строк), реализующим только одну функцию задачи, имеющим одну входную и одну выходную точку».

Первым основные свойства программного модуля более-менее четко сформулировал Парнас (Parnas): «Для написания одного модуля должно быть достаточно **минимальных** знаний о тексте другого». Таким образом, в соответствии с определением, модулем могла быть любая отдельная процедура (функция) как самого нижнего уровня иерархии (уровня реализации), так и самого верхнего уровня, на котором происходят только вызовы других процедур-модулей.

Таким образом, Парнас первым выдвинул концепцию скрытия информации (information hiding) в программировании. Однако существовавшие в языках 70-х годов только такие синтаксические конструкции, как процедура и функция, не могли обеспечить надежного скрытия информации, поскольку подвержены влиянию глобальных переменных, поведение которых в сложных программах бывает трудно предсказуемым.

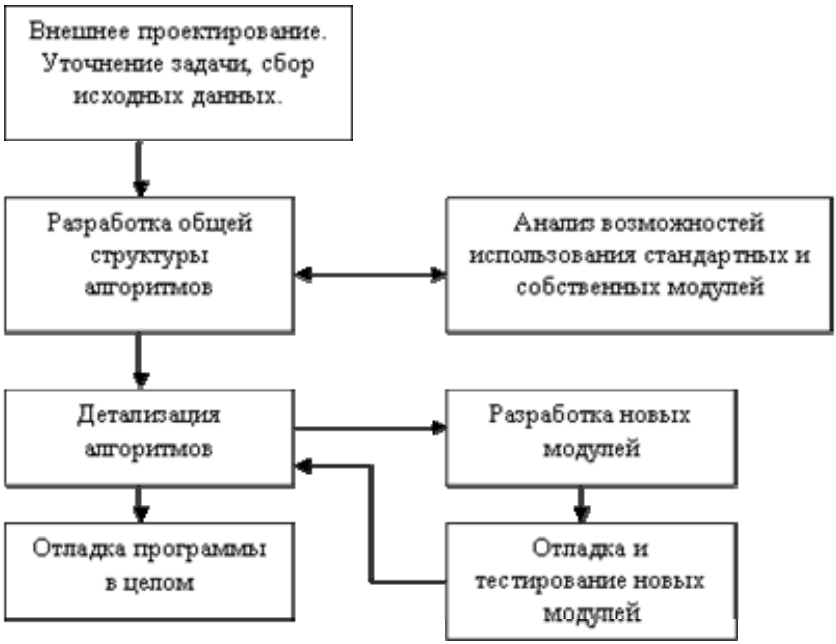
Решить эту проблему можно было только разработав новую синтаксическую конструкцию, которая не подвержена влиянию глобальных переменных.

Такая конструкция была создана и названа модулем. Изначально предполагалось, что при реализации сложных программных комплексов модуль должен использоваться наравне с процедурами и функциями как конструкция, объединяющая и надежно скрывающая детали реализации определенной подзадачи.

Таким образом, количество модулей в комплексе должно определяться декомпозицией поставленной задачи на независимые подзадачи. В предельном случае модуль может использоваться даже для заключения в него всего лишь одной процедуры, если необходимо, чтобы выполняемое ею локальное действие было гарантировано независимым от влияния других частей программы при любых изменениях.

Впервые специализированная синтаксическая конструкция модуля была предложена Н. Виртом в 1975 г. и включена в его новый язык Modula . Насколько сильно изменяются свойства языка, при введении механизма модулей, свидетельствует следующее замечание Н.Вирта, сделанное им по поводу более позднего языка Модула-2: «Модули – самая важная черта, отличающая язык Модула-2 от его предшественника Паскаля».

По своей организации и характеру использования в программе модули Паскаля близки к модулям-пакетам (PACKAGE) языка программирования Ада. В них так же, как и в пакетах Ады, явным образом выделяется некоторая «видимая» интерфейсная часть, в которой сконцентрированы описания глобальных типов, констант, переменных, а также приводятся заголовки процедур и функций. Появление объектов в интерфейсной части делает их доступными для других модулей и основной программы. Тела процедур и функций располагаются в исполняемой части модуля, которая может быть скрыта от пользователя.



Значение модулей для технологии разработки программного проекта может быть продемонстрировано диаграммой на рис. 2.

Модули представляют собой прекрасный инструмент для разработки библиотек прикладных программ и мощное средство модульного программирования. Важная особенность модулей заключается в том, что компилятор размещает их программный код в отдельном сегменте памяти. Длина сегмента не может превышать 64 Кбайт, однако количество одновременно используемых модулей ограничивается лишь доступной памятью, что позволяет создавать большие программы.

Структура модулей Паскаля

Всякий модуль Паскаля имеет следующую структуру:

Unit <имя_модуля>;
interface <интерфейсная часть>;
implementation < исполняемая часть >;
begin

<инициирующая часть>;
end .

Здесь UNIT – зарезервированное слово (единица); начинает заголовок модуля;

- <имя_модуля> - имя модуля (правильный идентификатор);
- INTERFACE – зарезервированное слово (интерфейс); начинает интерфейсную часть модуля;
- IMPLEMENTATION – зарезервированное слово (выполнение); начинает исполняемую часть модуля;
- BEGIN – зарезервированное слово; начинает инициирующую часть модуля; причем конструкция begin <инициирующая часть> необязательна;
- END – зарезервированное слово – признак конца модуля.

Таким образом, модуль Паскаля состоит из заголовка и трех составных частей, любая из которых может быть пустой.

Заголовок модуля Паскаля и связь модулей друг с другом

Заголовок модуля Паскаля состоит из зарезервированного слова unit и следующего за ним имени модуля. Для правильной работы среды Турбо Паскаля и возможности подключения средств, облегчающих разработку больших программ, имя модуля Паскаля должно совпадать с именем дискового файла, в который помещается исходный текст модуля. Если, например, имеем заголовок модуля Паскаля

Unit primer ;

то исходный текст этого модуля должен размещаться на диске в файле primer .pas .

Имя модуля Паскаля служит для его связи с другими модулями и основной программой. Эта связь устанавливается специальным предложением:

uses<список модулей>

Здесь **USES** – зарезервированное слово (использует);

<список модулей> - список модулей, с которыми устанавливается связь; элементы списка – имена модулей через запятую.

Если в Паскале модули используются, то предложение **uses <список модулей>** должно стоять сразу после заголовка программы , т.е. должно открывать раздел описаний основной программы. В модулях Паскаля могут использоваться другие модули. В модулях предложение uses <список модулей> может стоять сразу после слова interface или сразу после слова implementation . Допускается и два предложения uses , т.е. оно может стоять и там, и там.

Интерфейсная часть

Интерфейсная часть открывается зарезервированным словом **INTERFACE** . В этой части содержатся объявления всех глобальных объектов модуля (типов, констант, переменных и подпрограмм), которые должны быть доступны основной программе и (или) другим модулям Паскаля. При объявлении глобальных подпрограмм в интерфейсной части указывается только их заголовок, например:

Пример фрагмента программы

```
Unit complexn;
Interface Type Complex= record
  Re, im: real;
End;
Procedure AddC(x,y: complex, var z: complex);
Procedure MulC (x,y: complex, var z: complex);
```

Если теперь в основной программе написать предложение

Uses complexn ;

то в программе станут доступными тип cmplex и две процедуры – AddC и MulC из модуля complexn .

Отметим, что объявление подпрограмм в интерфейсной части автоматически сопровождается их компиляцией с использованием дальней модели памяти. Таким образом, обеспечивается доступ к подпрограммам из основной программы и других модулей Паскаля.

Следует учесть, что все константы и переменные, объявленные в интерфейсной части модуля Паскаля, равно как и глобальные константы и переменные основной программы, помещаются компилятором Турбо Паскаля в общий сегмент данных (максимальная длина сегмента 65536 байт).

Порядок появления различных разделов объявлений и их количество может быть произвольным. Если в интерфейсной части объявляются внешние подпрограммы или подпрограммы в машинных кодах, их тела (т.е. зарезервированное слово EXTERNAL , в первом случае, и машинные коды вместе со словом INLINE – во втором) должны следовать сразу за их заголовками в исполняемой части модуля (не в интерфейсной!). В интерфейсной части модулей Паскаля нельзя использовать опережающее описание.

Исполняемая часть модуля Паскаля

Исполняемая часть модуля Паскаля начинается зарезервированным словом **IMPLEMENTATION** и содержит описания подпрограмм, объявленных в интерфейсной части. В ней могут объявляться локальные для модуля объекты – вспомогательные типы, константы, переменные и блоки, а также метки.

Описанию подпрограммы, объявленной в интерфейсной части модуля Паскаля, в исполняемой части должен предшествовать заголовок, в котором можно опустить список формальных параметров и тип результата для функции, так как они уже описаны в интерфейсной части. Но если заголовок подпрограммы приводится в полном виде, т.е. со списком параметров и объявлением типа результата для функции, то он должен полностью совпадать с заголовком подпрограммы в интерфейсной части, например:

Пример модуля Паскаля

```
Unit complexn;
{-------------------------------}
Interface
Type
```

```
Complex= record
  Re, im: real;
End;
Procedure AddC(x,y: complex, var z: complex);
{-------------------------------}
Implementation
  Procedure AddC;
    z.re:= x.re + y.re;
    z.im:= x.im + y.im;
  end ;
end .
```

Инициирующая часть модуля Паскаля

Инициирующая часть завершает модуль Паскаля. Она может отсутствовать вместе с начинающим ее словом BEGIN или быть пустой – тогда вслед за BEGIN сразу следует признак конца модуля.

В инициирующей части размещаются исполняемые операторы, содержащие некоторый фрагмент программы. Эти операторы исполняются до передачи управления основной программе и обычно используются для подготовки ее работы. Например, в инициирующей части могут инициироваться переменные, открываться файлы, устанавливаться связи с другими компьютерами и т.п.:

Пример модуля Паскаля

```
Unit fileText;
{-----------------------------}
Interface
  Procedure print(s: string);
{-----------------------------}
implementation
var f: text;
const
  name= ‘output.txt’;
procedure print;
begin
```

```
  writeln(f, s)
end ;
{--------------------------------}
{начало инициирующей части}
begin
  assign(f, name);
  rewrite ( f );
{конец инициирующей части}
end .
```

Не рекомендуется делать инициирующую часть пустой, лучше ее опустить.

Компиляция модулей Паскаля

В среде Турбо Паскаль имеются средства, управляющие способом компиляции модулей и облегчающие разработку больших программ. Определены три режима компиляции: **COMPILE , MAKE , BUILD**. Режимы отличаются способом связи компилируемого модуля или основной программы с другими модулями, объявленными в предложении USES .

При компиляции модуля или основной программы в режиме **COMPILE** все, упоминаемые в предложении USES модули, должны быть предварительно откомпилированы, и результаты компиляции должны быть помещены в одноименные файлы с расширением TPU (от англ. Turbo Pascal Unit). Файл с расширением TPU создается автоматически при компиляции модуля Паскаля.

В режиме **MAKE** компилятор проверяет наличие TPU -файлов для каждого объявленного модуля. Если какой-либо файл не найден, система ищет одноименный файл с расширением PAS , т.е. файл с исходным текстом модуля Паскаля. Если таковой файл найден, система приступает к его компиляции. Кроме того, в этом режиме система следит за возможными изменениями исходного текста любого используемого модуля. Если в PAS -файл внесены изменения, то независимо от того, есть ли в каталоге соответствующий TPU -файл или нет, система откомпилирует его перед

компиляцией основной программы. Более того, если изменения внесены в интерфейсную часть, то будут откомпилированы все другие модули, обращающиеся к нему. Режим MAKE существенно облегчает процесс разработки крупных программ с множеством модулей Паскаля: программист избавляется от необходимости следить за соответствием TPU -файлов их исходному тексту, т.к. система делает это автоматически.

В режиме **BUILD** существующие TPU -файлы игнорируются, система пытается отыскать и откомпилировать соответствующие PAS - файлы для каждого модуля Паскаля. После компиляции можно быть уверенным, что учтены все сделанные в текстах модулей Паскаля исправления и изменения.

Подключение модулей Паскаля к основной программе и их компиляция происходит в порядке их объявления в предложении USES . При переходе к очередному модулю Паскаля система предварительно ищет все модули, на которые он ссылается. Ссылки модулей Паскаля друг на друга могут образовывать древовидную структуру любой сложности, однако запрещается явное или косвенное обращение модуля к самому себе. Например, недопустимы следующие объявления:

Пример ошибок модуля Паскаля

```
Unit A; Unit B;
interface interface
uses B;Uses A;
……. ……
implementation implementation
…….. ……
end. end.
```

Это ограничение можно обойти, если «спрятать» предложение USES в исполняемые части зависимых модулей:

Пример исправленных ошибок модуля Паскаля

```
Unit A; Unit B;
interface interface
```

……. ……
implementation implementation
uses B;Uses A;
…….. ……
end. end.

Дело в том, что Турбо Паскаль разрешает ссылки на частично откомпилированные модули, что приблизительно соответствует опережающему описанию подпрограммы. Если интерфейсные части независимы (это обязательное условие!), Турбо Паскаль сможет идентифицировать все глобальные объекты в каждом модуле, после чего откомпилирует тела модулей обычным способом.

Доступ к объявленным в модуле Паскаля объектам

Пусть, например, мы создаем модуль Паскаля, реализующий сложение и вычитание комплексных чисел с помощью процедур:

Пример модуля реализующий сложение и вычитание комплексных чисел

```
Unit complexn;
Interface
  type
  complex= record
  re, im: real;
  end;
procedure AddC (x, y: complex; var z: complex);
procedure SubC (x, y: complex; var z: complex);
const c: complex= (re: 0.1; im: -1);
implementation
procedure AddC;
begin
  z.re:= x.re + y.re;
  z.im:= x.im + y.im;
end; {AddC}
procedure SubC;
begin
```

```
  z.re:= x.re - y.re;
  z.im:= x.im - y.im;
end; {SubC}
end .
```

Текст этого модуля следует поместить в файл complexn . pas . Вы можете его откомпилировать, создав TPU -файл.

В следующей программе осуществляются арифметические операции над комплексными числами:

Арифметические операции над комплексными числами

```
    Program primer ;
Uses complexn;
Var
  a,b,c: coplex;
begin
  a.re:= 1; a.im:= 1;
  b.re:= 1; b.im:= 2;
  AddC(a, b, c);
  Writeln (‘ сложение :’, c.re: 5:1, c.im: 5:1, ‘i’);
  SubC (a, b, c);
  Writeln (‘ вычитание :’, c.re: 5:1, c.im: 5:1, ‘i’);
End.
```

После объявления Uses complexn программе стали доступны все объекты, объявленные в интерфейсной части модуля complexn . При необходимости можно переопределить любой из этих объектов, как произошло, например, с типизированной константой ***c*** , объявленной в модуле Паскаля. Переопределение объекта означает, что вновь объявленный объект «закрывает» ранее определенный в модуле одноименный объект. Чтобы получить доступ к «закрытому» объекту, нужно воспользоваться составным именем: перед именем объекта поставить имя модуля и точку. Например :

```
    Writeln (complexn.c.re: 5: 1, complexn.c.im: 5: 1);
```

Этот оператор выведет на экран содержимое «закрытой» типизированной константы, объявленной в модуле Паскаля из предыдущего примера.

Стандартные модули Паскаля

В Турбо Паскале имеется 8 стандартных модулей, в которых содержится множество различных типов, констант, процедур и функций. Этими модулями являются SYSTEM, DOS, CRT, GRAPH, OVERLAY, TURBO3, GRAPH3. Модули Паскаля GRAPH , TURBO 3, GRAPH 3 выделены в отдельные TPU -файлы, а остальные входят в состав библиотечного файла TURBO . TPL . Лишь один модуль Паскаля SYSTEM подключается к любой программе автоматически, все остальные становятся доступны только после указания их имен в списке подключаемых модулей.

Модуль Паскаля SYSTEM. В него входят все процедуры и функции стандартного Паскаля, а также встроенные процедуры и функции, которые не вошли в другие стандартные модули (например, INC , DEC , GETDIR и т.п.). Модуль Паскаля SYSTEM подключается к любой программе независимо от того, объявлен ли он в предложении USES или нет, поэтому его глобальные константы, переменные, процедуры и функции считаются встроенными в Турбо Паскаль.

Модуль Паскаля PRINTER делает доступным вывод текстов на матричный принтер. В нем определяется файловая переменная LST типа TEXT , которая связывается с логическим устройством PRN. После подключения данного модуля Паскаля можно выполнить, например, такое действие:

Пример стандартного модуля Паскаля

```
Uses printer;
Begin
  Writeln(lst, ‘ Турбо Паскаль ’);
End.
```

Модуль Паскаля CRT. В нем сосредоточены процедуры и функции, обеспечивающие управление текстовым режимом

работы экрана. С его помощью можно перемещать курсор в любую точку экрана, менять цвет выводимых символов и фона, создавать окна. Кроме того, в данный модуль включены также процедуры «слепого» чтения клавиатуры и управления звуком.

Модуль Паскаля GRAPH . Содержит набор типов, констант, процедур и функций для управления графическим режимом работы экрана. Этот модуль позволяет создавать различные графические изображения и выводить на экран надписи стандартными или созданными программистом шрифтами.

Модуль Паскаля DOS . В модуле собраны процедуры и функции, открывающие доступ к средствам дисковой операционной системы MS - DOS .

Модуль Паскаля OVERLAY . Данный модуль необходим при разработке громоздких программ с перекрытиями. Турбо Паскаль обеспечивает создание программ, длина которых ограничивается лишь основной оперативной памятью. Операционная система MS - DOS оставляет программе около 580 Кбайт основной памяти. Память такого размера достаточна для большинства исполняемых программ, тем не менее, использование программ с перекрытиями снимает это ограничение.

Модули Паскаля TURBO 3 и GRAPH 3 введены для обеспечения совместимости с ранней версией системы Турбо Паскаль.

Задание: создать личный модуль, содержащий указанные в варианте курсовой подпрограммы; написать программу, которая подключает данный модуль и использует подпрограммы.

Список вариантов курсовой работы

Вариант № 1

1. Записать подряд в массивы Y положительные, а в массив Z отрицательные элементы массива $x_1, x_2, \dots, x_{20}$.
2. Вычислить произведение двух прямоугольных матриц.
3. В матрице A (m×n) подсчитать произведение элементов, расположенных ниже главной диагонали.

Вариант № 2

1. Выбрать из натурального ряда и поместить в целочисленный массив A (n) в порядке возрастания сначала единицу, а затем все числа, кратные шести.
2. Вычислить след матрицы A (n×n) (сумму элементов главной диагонали).
3. Из матрицы A (n×n) вычеркнуть K-й столбец. K ≤ n.

Вариант № 3

1. В матрице A (m×n) столбцы, в которых есть хотя бы один отрицательный элемент, заменить нулями.
2. В матрице A (m×n) найти столбцы, в которых на четных местах стоят только положительные элементы.
3. Вычислить $K!$.

Вариант № 4

1. В матрице A (m×n) элементы K-го столбца заменить суммой положительных элементов каждой строки.
2. В матрице A (m×n) определить номер строки и столбца первой встречающейся единицы.

3. Вычислить a^n, где n – целое число.

Вариант № 5

1. Для массива y (9) вычислить

$$z = \sum_{i=1}^{9} y_i \prod_{a=1}^{5} \cos(a + 2i)$$

2. В матрице A (m×n) все элементы, равные единице, заменить суммой положительных элементов той строки, где встретилась первая единица.
3. Для массива X (n) найти сумму элементов, расположенных между максимальным и минимальным элементами. В сумму включить эти элементы.

Вариант № 6

1. Получить n-е число Фибоначчи. Числа Фибоначчи определяются по следующему правилу: $f_1 = 1,\ f_2 = 1,\ f_{i+1} = f_i + f_{i-1}$, для i ≥ 2.
2. В матрице A (m×n) расположить строки в порядке возрастания сумм их элементов.
3. В матрице A (m×n) поменять местами элементы K и K+1 строки ($K + 1 \leq m$)

Вариант № 7

1. Дана матрица A (m×n). Получить массив B (n), элементы которого представляют собой разности между максимальным элементом матрицы A и минимальным элементом каждого столбца.
2. Вычислить значение функции

$$z = \sum_{i=1}^{20} \prod_{a}^{b} \frac{x_i + y_k}{h},$$

где x_i - элемент массива $x_1, x_2, \dots, x_{20}$, y_k изменяется от a до b с шагом h.

3. Разделить матрицу на число.

Вариант № 8

1. В матрице A (m×n) определить максимальное значение суммы строки и номер этой строки.
2. Для вещественной матрицы A (m×n) подсчитать произведение элементов, расположенных выше побочной диагонали.
3. Найти первые n простых чисел.

Вариант № 9

1. Из вещественной матрицы A (m×n) вычеркнуть k-ю строку ($k \le m$).
2. Из элементов столбца с номером k массива A (m×n) вычесть минимальный элемент этого столбца.
3. Для матрицы A (m×n) найти произведение положительных элементов побочной диагонали.

Вариант № 10

1. В матрице A (m×n) определить номер и величину того элемента, который меньше всего отличается от максимального элемента и не равен ему.
2. Получить сумму двух прямоугольных матриц.
3. Вычислить скалярное произведение векторов X (n) и Y (n).

Вариант № 11

1. В матрице A (m×n) расположить строки по убыванию произведений их отрицательных элементов.
2. В массиве D (n) сумму отрицательных элементов поставить на место минимального элемента.
3. Для матрицы A (m×n) получить произведение отрицательных элементов столбцов с четными номерами.

Вариант № 12

1. Максимальный элемент главной диагонали матрицы A (n×n) заменить суммой положительных элементов строки с номером K ($k \le n$).
2. В матрице A (m×n) минимальные элементы каждой строки заменить порядковым номером этого элемента в строке.
3. Вычислить для действительного числа X и массива y (k), $k \ge 20$ значение функции

$$S = \sum_{k=1}^{20} y_k \prod_{i=1}^{10} \sin(ix + k).$$

Вариант № 13

1. В матрице A (m×n) каждую строку разделить на минимальный элемент этой строки.
2. Из матрицы A (m×n) вычеркнуть столбец и строку, на пересечении которых находится максимальный элемент.
3. Вычислить $\arcsin x$,

$$\arcsin x = \operatorname{arctg}\frac{x}{\sqrt{1-x^2}}.$$

Вариант № 14

1. В матрице A (m×n) определить минимальный элемент выше главной диагонали.
2. В матрице A (m×n) расположить строки по возрастанию сумм их элементов.
3. Найти разность двух прямоугольных матриц.

Вариант № 15

1. Дана матрица A (m×n). Получить массив B (m), элементы которого представляют собой среднее арифметическое положительных элементов строки.
2. Для матрицы A (m×n) элементы главной диагонали расположить в порядке убывания.
3. В матрице A (m×n) поменять местами k и k+1 столбцы, $k+1 \le n$.

Вариант № 16

1. Вычислить определить матрицы A (n×n).
2. В матрице A (m×n) расположить столбцы в порядке убывания сумм их элементов.
3. Вычислить произведение элементов, расположенных ниже побочной диагонали матрицы A (m×n).

Вариант № 17

1. Умножить две прямоугольные матрицы.
2. Вычислить детерминант матрицы A (n×n).

3. В матрице A (n×n) подсчитать произведение положительных элементов выше главной диагонали.

Вариант № 18

1. В матрице B (M,N) расположить строки по убыванию их отрицательных элементов.
2. В матрице A (n×n) подсчитать сумму положительных элементов, расположенных выше главной диагонали, и их количество.
3. В матрице B (m×m) подсчитать произведение элементов, расположенных на побочной диагонали.

Вариант № 19

1. В матрице B (m,m) подсчитать произведение элементов, расположенных на побочной диагонали.
2. Вычислить $\arcsin x$,
$$\arcsin x = \operatorname{arctg}\frac{x}{\sqrt{1-x^2}}.$$
3. В массиве D (n) сумму отрицательных элементов поставить на место минимального элемента.

Вариант № 20

1. Дана матрица A (m×n). Получить массив B (m), элементы которого представляют собой среднее арифметическое положительных элементов строки.
2. Найти первые n простых чисел.
3. Найти разность двух прямоугольных матриц.

Вариант № 21

1. В матрице A (m×n) определить минимальный элемент выше главной диагонали.

2. Вычислить определитель матрицы A (4×4).
3. Вычислить $\arccos x$,

$$\arccos x = \operatorname{arctg}\frac{\sqrt{1-x^2}}{x}.$$

Вариант № 22

1. В матрице A (m×n) столбцы, в которых есть хотя бы один отрицательный элемент, заменить нулями.
2. В матрице A (m×n) подсчитать сумму элементов, расположенных выше главной диагонали.
3. В матрице A (m×n) подсчитать произведение элементов, расположенных ниже главной диагонали.

Вариант № 23

1. В матрице A (m×n) найти столбцы, в которых на четных местах стоят только положительные элементы.
2. Для массива y (9) вычислить

$$z = \sum_{i=1}^{9} y_i \prod_{a=1}^{5} \cos(a + 2i).$$

3. В матрице A (m×n) поменять местами элементы K и K+1 строки ($K + 1 \leq m$).

Вариант № 24

1. Для вещественной матрицы A (m×n) подсчитать произведение элементов, расположенных выше побочной диагонали.

2. В матрице A (m×n) определить номер и величину того элемента, который меньше всего отличается от максимального элемента и не равен ему.
3. Для матрицы A (m×n) получить произведение отрицательных элементов столбцов с четными номерами.

Вариант № 25

1. Из матрицы A (m×n) вычеркнуть столбец и строку, на пересечении которых находится максимальный элемент.
2. Дана матрица A (m×n). Получить массив B (m), элементы которого представляют собой среднее арифметическое положительных элементов строки.
3. Вычислить произведение элементов, расположенных ниже побочной диагонали матрицы A (m×n).

Вариант № 26

1. В матрице A (n×n) подсчитать сумму положительных элементов, расположенных выше главной диагонали, и их количество.
2. Дана матрица A (m×n). Получить массив B (m), элементы которого представляют собой среднее арифметическое положительных элементов строки.
3. Вычислить $\arccos x$,

$$\arccos x = \operatorname{arctg}\frac{\sqrt{1-x^2}}{x}.$$

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Акулов О.А. Информатика : базовый курс: учебник .— 5-е изд., испр. и доп. — М. : ОМЕГА-Л, 2008 .— 574 с.
2. Вислова Е.В. Информатика. Турбо Паскаль: Задачник: Учеб. пособие. — Воронеж: ВАИУ, 2008. - 80 с.

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения курсовой работы по дисциплине
«Алгоритмические языки и программирование»
для студентов специальности 160700.65, 24.05.02
«Проектирование авиационных и
ракетных двигателей» очной формы обучения

Составители: Шматов Дмитрий Павлович
Сушков Алексей Михайлович

В авторской редакции

ФГБОУ ВПО «Воронежский государственный технический
университет»
394026 Воронеж, Московский пр., 14