***Массивы. Основные алгоритмы обработки массивов на примере языка программирования Pascal***

**Оглавление**

[Введение 3](#_Toc217970559)

[1. Определение и типы массивов 4](#_Toc217970560)

[2. Основные операции обработки массивов 7](#_Toc217970561)

[2.1 Определение размерности массива, заполнение массива 7](#_Toc217970562)

[2.2 Вывод массива на экран 9](#_Toc217970563)

[2.3 Поиск требуемого элемента в массиве 10](#_Toc217970564)

[2.4 Поиск максимального и минимального элементов массива 12](#_Toc217970565)

[2.5 Сортировка элементов массива 13](#_Toc217970566)

[3. Особенности обработки двумерных массивов 15](#_Toc217970567)

[4. Обработка квадратных матриц 17](#_Toc217970568)

[4.1 Определение диагоналей массива 17](#_Toc217970569)

[4.2 Определение четвертей матрицы 18](#_Toc217970570)

[5. Открытые массивы 20](#_Toc217970571)

[Список литературы 21](#_Toc217970572)

**Введение**

Тема данного реферата «Массивы. Основные алгоритмы обработки массивов на примере языка программирования Pascal». Актуальность выбранной темы обусловлена тем, что массивы очень широко используются при разработке различного рода приложений. Массивы являются распространенным и полезным способом сохранения многих различных частей связанных данных. Массивы полезны при создании отсортированных и неотсортированных списков данных, при сохранении таблиц данных и для выполнения многих других задач. С понятием «массив» приходится работать и при решении научно-технических и экономических задач, связанных с обработкой совокупностей большого количества значений.

Массив является удобным способом хранения нескольких связанных элементов данных в едином контейнере для большего удобства и эффективности программирования. Массив позволяет сохранять и манипулировать многими элементами данных посредством единственной переменной. Кроме уменьшения общего числа различных имен переменных, которые необходимо отслеживать, другим основным преимуществом использования массивов является то, что можно использовать циклы для легкой обработки различных элементов массивов. Объединяя массивы и циклы можно написать небольшое число операторов, которые обрабатывают большой объем данных. Выполнение тех же задач с использованием отдельных переменных может потребовать написания сотен операторов.

В связи со всем изложенным выше цель данной работы – рассмотреть основные алгоритмы обработки массивов максимально близко к практическому их применению.

**1.** **Определение и типы массивов**

Массив - это множество однотипных элементов, объединённых общим именем и занимающих в компьютере определённую область памяти. Количество элементов в массиве всегда конечно. В общем случае массив - это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.

Другими словами можно сказать, что массив представляет собой фиксированное количество упорядоченных однотипных компонент, снабженных индексами, т.е. является совокупностью конечного числа данных одного типа. В качестве элементов массива можно использовать любой тип данных, поэтому вполне правомерно существование массивов записей, массивов указателей, массивов строк, массивов и т.д.

Массивы могут быть:

* одномерными (одна строка – несколько столбцов);
* многомерными (несколько строк – несколько столбцов).

Для создания массива его предварительно необходимо описать либо в разделе var, либо в разделе type. Для задания массива используется зарезервированное слово **array**, после которого указывается тип индекса (-ов) компонент (в квадратных скобках) и после слова **of -** тип самих компонент:

**Type**

**<имя массива>=array[<тип индекса(-ов)>] of <тип компонент>;**

Или

**Var**

**<имя массива>:array[<тип индекса(-ов)>] of <тип компонент>;**

Введя тип массив, можно задавать переменные или типизированные константы этого типа. Размерность массива может быть любой, компоненты массива могут быть любого, в том числе и структурированного, типа; индекс может быть любого порядкового типа, кроме типа longint.

При задании значений константе-массиву компоненты указываются в круглых скобках и разделяются запятыми, причем, если массив многомерный, внешние круглые скобки соответствуют левому индексу, вложенные в них круглые скобки - следующему индексу и т.д.

**Например,**

Type

arr = array [1..3] of real;

matrix = array [1..3, 1..2] of integer;

Const

3

4

5

6

массив mas2

1

2

mas1: arr = (1, 2, 3);

mas2: matrix = ((1, 2), (3, 4), (5, 6));

Тип массив можно вводить и непосредственно при определении соответствующих переменных или типизированных констант.

**Например,**

Var

m1, m2 : array [1..3] of integer;

matr : array [1..3, 1..3] of real;

Доступ к компонентам массива осуществляется указанием имени массива, за которым в квадратных скобках помещается значение индекса (-ов) компоненты. В общем случае каждый индекс компоненты может быть задан выражением соответствующего типа.

**Например,** m1 [2], matr[i,j].

Для обработки массива и последовательного доступа к данным, как правило, используется цикл FOR.

**Например,**

for i:=1 to 10 do read(mas[i]);

Обработка элементов двумерного массива обычно выполняется с помощью двойного цикла. Один цикл управляет перебором номеров строк, другой - столбцов.

**Например,**

for i:=1 to 10 do

for j:=1 to 10 do read(mas[i, j]);

Над элементами массива можно производить те же операции, которые допустимы для данных его базового типа. Если два массива имеют одинаковые типы индексов и одинаковые типы элементов, то к ним применимы булевы операции (<>=).

**2.** **Основные операции обработки массивов**

**2.1 Определение размерности массива, заполнение массива**

Размерностью массива называется количество ячеек данного массива. При описании массива необходимо указывать конкретное число ячеек, но при реализации задачи не всегда необходимо заполнять все зарезервированные ячейки.

**Например,**

**Задача 1.** «Дан одномерный массив из 10 компонент...» - эта формулировка означает, что и при описании и при обработке массива всегда будут использоваться 10 ячеек.

**Задача 2.** «Дан массив размерность N…» - данная формулировка означает, что размерность массива будет определяться самим пользователем. Т.е. от разработчика такой программы требуется:

* определить максимальную размерность массива (как правило, вполне достаточно 100 ячеек);
* дать возможность пользователю указать количество требуемых ячеек (writeln(“Введите размерность массива”);

readln(n)- теперь n обозначает размерность).

Для того чтобы заполнить массив необходимо последовательно перебрать все ячейки (компоненты) массива и записать в них некоторые значения. Для перебора ячеек массива используется цикл **for**, в котором с помощью счетчика перебираются индексы ячеек. Заносимые в массив данные могут запрашиваться как с клавиатуры, так и выбираться случайным образом с помощью генератора случайных чисел Random.

**Например,**

**Способ 1.** Заполнение одномерного массива с клавиатуры

writeln(“Введите размерность массива”);

readln(n);

For i:=1 to n do

Begin

Writeln(“Введите ”,i,” элемент массива”);

Readln(mas[i]);

End;

**Ход выполнения:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **i =** | **Writeln** | **Readln** | **Действие** |
| **Шаг 1** | 1 | Введите 1 элемент массива | Mas[1] | Считываем число в 1 ячейку |
| **Шаг 2** | 2 | Введите 1 элемент массива | Mas[2] | Считываем число в 2 ячейку |
| **Шаг 3** | 3 | Введите 1 элемент массива | Mas[3] | Считываем число в 3 ячейку |
| **…** | **…** | **…** | **…** | **…** |
| **Шаг n** | n | Введите n элемент массива | Mas[n] | Считываем число в последнюю ячейку |

**Способ 2.** Заполнение массива случайными числами

writeln(“Введите размерность массива”);

readln(n); {определяем размерность массива}

Randomize; {включаем генератор случайных чисел}

For i:=1 to n do {начинаем перебирать массив}

Mas[i]:=random(100) {выбор любого числа из указанного диапазона и размещение его в массиве}

**Ход выполнения:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **i =** | **Random** | **Mas[i]** | **Действие** |
| **Шаг 1** | 1 | Любое число до 100 | Любое число до 100 | Считываем выбранное число в массив |
| **Шаг 2** | 2 | Любое число до 100 | Любое число до 100 | Считываем число в 2 ячейку |
| **Шаг 3** | 3 | Любое число до 100 | Любое число до 100 | Считываем число в 3 ячейку |
| **…** | **…** |  |  | **…** |
| **Шаг n** | n | Любое число до 100 | Любое число до 100 | Считываем число в последнюю ячейку |

***Примечание:*** *в случае, если размерность массива известна заранее, то чикл for выполняется от 1 до определенного числа.*

***Например****, for i:=1 to 5 do – перебор 5 ячеек массива.*

**2.2 Вывод массива на экран**

Для вывода массива необходимо последовательно перебрать все ячейки (компоненты) массива и вывести лежащие там значения на экран с помощью оператора **write / writeln**.

Оператор **write** выведет значения массива в строку

Оператор **writeln** выведет значения массива в столбец

Для перебора ячеек массива используется цикл **for**, в котором с помощью счетчика перебираются индексы ячеек.

**Способ 1**. Вывод одномерного массива размерностью 3 с помощью оператора writeln

Writeln(‘Массив’);

For i:=1 to 3 do

Writeln(mas[i]);

Массив

1

2

3

Экранное представление

**Способ 2.** Вывод одномерного массива размерностью 3 с помощью оператора write

Writeln(‘Массив:’);

For i:=1 to 3 do

Write(mas[i],’ ‘);

Массив

1 2 3

Экранное представление

**2.3 Поиск требуемого элемента в массиве**

Общий алгоритм поиска в массиве определенного элемента можно представить следующим образом:

Количество повторов = размерность массива

**-**

**+**

«зайти» в ячейку с опред. индексом

Содержимое = искомое ?

Требуемые действия

Увеличиваем индекс ячейки

Рисунок 1. Блок-схема поиска требуемого элемента в массиве

**Например,**

Дан одномерный массив из 7 ячеек. Определить, сколько в нем чисел кратных 7.

Var

mas:array[1..7] of integer;

i:integer; {необходима для перебора массива}

kol:integer; {количество подходящих элементов}

begin

for i:=1 to 7 do

begin

write(‘Введите’ ,i, ‘ элемент’);

readln(mas[i]); {заполняем массив}

if (mas[i] mod 7 =0) and (mas[i]<>0) then kol:=kol+1

{если элемент делится на 7 и в остатке ноль и при этом сам элемент не равен нулю, то увеличиваем счетчик kol}

end;

writeln(“Количество чисел кратных 7 -”, kol) {вывод результата}

end.

**Ход выполнения:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | -7 | 0 | 14 | 5 | 21 | -5 | Элементы |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | Индексы |

Введенный массив

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **i =** | **Readln(mas[i])** | **Проверка**  **(mas[i] mod 7 =0) and (mas[i]<>0)** | **Действие** |
| **Шаг 1** | 1 | Mas[1]=9 | ложь | Kol=0 |
| **Шаг 2** | 2 | Mas[2]=-7 | истина | Kol=0+1 |
| **Шаг 3** | 3 | Mas[3]=0 | ложь | Kol=1 |
| **Шаг 4** | 4 | Mas[4]=14 | истина | Kol=1+1 |
| **Шаг 5** | 5 | Mas[5]=5 | ложь | Kol=2 |
| **Шаг 6** | 6 | Mas[6]=21 | истина | Kol=2+1 |
| **Шаг 7** | 7 | Mas[7]= -5 | ложь | Kol=3 |
| Вывод результата на экран | | | | |

**2.4 Поиск максимального и минимального элементов массива**

Общий алгоритм работы можно представить в следующем виде:

* определяются переменные (max и min), в которых в результате выполнения алгоритма разместятся соответственно максимальное и минимальное значения;
* начинается перебор массива с одновременной проверкой «является ли просматриваемый элемент больше максимального или меньше минимального»; в случае, если условие истинно – значение элемента массива присваивается соответствующей переменной;

Основной алгоритм выглядит следующим образом:

Max:= - 32000;

Min:= 32000;

For i:=1 to n do

begin

If mas[i]>max then max:=mas[i];

If mas[i]<min then min:=mas[i];

End;

**Ход выполнения:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | -7 | 0 | -10 | Элементы |
| 1 | 2 | 3 | 4 | Индексы |

Введенный массив

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **i =** | **Данные** | **Условие** | **Ответ** | **Действие** |
| **Шаг 1** | 1 | Max=-32000  Min=32000  Mas[1] = 9 | 9 > -32000 9 < 32000 | истина  истина | Max=9  Min=9 |
| **Шаг 2** | 2 | Max=9  Min=9  Mas[2]= - 7 | -7 > 9  -7 < 0 | ложь  истина | Max=9  Min= - 7 |
| **Шаг 3** | 3 | Max=9  Min= - 7  Mas[3]=0 | 0 > 9  0 < -7 | ложь  ложь | Max=9  Min= - 7 |
| **Шаг 4** | 4 | Max=9  Min= - 7  Mas[4] = - 10 | -10 > 9  - 10 < -7 | ложь  истина | Max=9  Min= - 10 |

**2.5 Сортировка элементов массива**

Существует много алгоритмов сортировки массива, но наиболее простым и понятным является сортировка методом «пузырька», при которой самый «легкий» элемент «всплывает», а самый тяжелый «тонет».

**Например,**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | -7 | 0 | -8 | Элементы |
| 1 | 2 | 3 | 7 | Индексы |

Дан массив

Необходимо расположить эти элементы в порядке возрастания, т.е. в результате работы программы необходимо получить массив

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -8 | -7 | 0 | 9 | Элементы |
| 1 | 2 | 3 | 7 | Индексы |

При сортировке методом «пузырька» сравниваются два соседних элемента (mas[i] и mas[i+1]). Если mas[i] > mas[i+1], то происходит перестановка элементов. Визуально процесс сортировки можно представить в виде:

Процесс повторяется N раз, где N – размерность массива

9 >-8

истина

меняем местами

9 >0

истина

меняем местами

mas[i+1]

mas[i]

9 >-7

истина

меняем местами

Самый тяжелый элемент «упал»

Рисунок 2. Сортировка методом «пузырька»

Таким образом, для организации сортировки потребуется два цикла, которые выстраиваются в следующем порядке:

(n-размерность массива)

For j:=1 to n do {количество просмотров массива}

For i:=1 to n-1 do {перебор элементов массива}

If mas[i] > mas[i+1] then begin

перестановка

элементов

массива

t:=mas[i];

mas[i]:=mas[i+1];

mas[i+1]:=t;

end;

Как видно из примера для перестановки элементов массива требуется дополнительная переменная, которая служит временным хранилищем значения ячейки массива.

t = 9

получаем

t = 9

получаем

t = 9

получаем

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | -7 | 0 | -8 |

**Шаг 1.**

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | -7 | 0 | -8 |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | -7 | 0 | -8 |

|  |  |  |  |
| --- | --- | --- | --- |
| -7 | -7 | 0 | -8 |

**Шаг 2.**

|  |  |  |  |
| --- | --- | --- | --- |
| -7 | -7 | 0 | -8 |

|  |  |  |  |
| --- | --- | --- | --- |
| -7 | 9 | 0 | -8 |

**Шаг 3.**

**3. Особенности обработки двумерных массивов**

**Двумерный массив –** структура данных, хранящая в себе прямоугольную матрицу. В матрице каждый элемент определяется номером строки и номером столбца, на пересечении которых он расположен.

Для описания двумерных массивов используются те же способы, что и для одномерных массивов, но в качестве размерности массива задается двойное значение (Например, ***[1..10,1..10]***).

Таким образом, для создания двумерного целочисленного массива размерностью 5×7 (5 строк, 7 столбцов) необходимо записать:

**Способ 1**

Type mas=array[1..5,1..7] of integer;

**Способ 2**

Var mas:array[1..5,1..7] of integer;

Для последовательного перебора всех элементов двумерного массива необходимо использовать т.н. вложенный цикл:

For i:=1 to 5 do {перебор строк матрицы}

For j:=1 to 7 do {перебор столбцов (ячеек) в строке}

Т.е. значение индекса строки (i) увеличится только в том случае, если индекс столбца (j) дойдет до своего конечного значения (в примере j = 7).

При такой организации перебора элементов массива процесс перебора будет проходить по следующей схеме:

11, 12 … - индексы ячеек

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 |

Рисунок 3. Процесс перебора элементов двумерного массива

**Ход выполнения:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **i =** | **j=** | **Условие перехода на следующую строку j=7** | **Обрабатываемый элемент** |
| **Шаг 1** | 1 | 1 | нет | Mas[1,1] |
| **Шаг 2** | 1 | 2 | нет | Mas[1,2] |
| **Шаг 3** | 1 | 3 | нет | Mas[1,3] |
| **Шаг 4** | 1 | 4 | нет | Mas[1,4] |
| **Шаг 5** | 1 | 5 | нет | Mas[1,5] |
| **Шаг 6** | 1 | 6 | нет | Mas[1,6] |
| **Шаг 7** | 1 | 7 | да | Mas[1,7] |
| **Шаг 8** | 2 | 1 | нет | Mas[2,1] |
| **Шаг 9** | 2 | 2 | нет | Mas[2,2] |
| **Шаг 10** | 2 | 3 | нет | Mas[2,3] |
| **…** | **…** | **…** | **…** | **…** |

Для того чтобы вывести двумерный массив на экран в виде таблицы необходимо после вывода содержимого каждой строки предусмотреть переход на строку ниже:

For i:=1 to n do

Begin

For j:=1 to n do

Write(mas[i,j],’ ‘);

Writeln;

End;

**4. Обработка квадратных матриц**

Если количество строк массива равно количеству столбцов, то такой массив называется квадратной матрицей. При работе с квадратной матрицей, в отличие от обычного двумерного массива, можно выделить:

* диагонали (главная, побочная);
* элементы, расположенные над и под диагоналями;
* четверти матрицы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 12 | 13 | 14 | 15 |
| 21 | 22 | 23 | 24 | 25 |
| 31 | 32 | 33 | 34 | 35 |
| 41 | 42 | 43 | 44 | 45 |
| 51 | 52 | 53 | 54 | 55 |

Здесь первая цифра номера элемента обозначает номер строки матрицы (i), вторая цифра – номер столбца (j)

Для определения элементов, входящих в любой из перечисленных разделов, существует формула, основными составляющими которой являются i – номер строки, j – номер столбца и N – размерность массива. Например, для определения элемента с номером 43, расположенного под побочной диагональю можно использовать формулу **i+j>N+1**, где i=4, j=3, N=5, таким образом, получаем 4+3>5+1.

**4.1 Определение диагоналей массива**

Побочная диагональ, **i + j = N+1**

Главная диагональ, **i = j**

Рисунок 4. Диагонали двумерного массива

Таким образом, в матрице, представленной в п. 4, элементы с номерами 11, 22, 33, 44 и 55 являются элементами главной диагонали. Элементы с номерами 15, 24, 33, 42, 51 – элементы побочной диагонали.

Расположение элементов, находящихся над или под диагональю, определяется по отношению к одной из диагоналей.

**i+j<N+1**

**i < j**

**i > j**

**i+j>N+1**

Рисунок 5. Расположение элементов по отношению к диагоналям

Элементы 12, 13, 14, 15, 23, 24, 25, 34, 35 и 45 расположены над главной диагональю, 21, 31, 32, 41, 42, 43, 51, 52, 53, 54 расположены под главной диагональю. Элементы 11, 12, 13, 14, 21, 22, 23, 31, 32 и 41 расположены над побочной диагональю, 25, 34, 35, 43, 44, 45, 54, 53, 54, 55 расположены под побочной диагональю.

**4.2 Определение четвертей матрицы**

Относительно обеих диагоналей элемент массива может находиться в одной из четвертей.

12, 13, 14, 23 – элементы первой четверти

25, 34, 35, 45 – элементы второй четверти

43, 52, 53, 54 – элементы третьей четверти

21, 31, 32, 41 – элементы четвертой четверти

IV

III

II

I

**(i < j) and (i + j < N + 1)**

**(i < j) and (i + j > N + 1)**

**(i > j) and (i + j < N + 1)**

**(i > j) and (i + j > N + 1)**

Рисунок 6. Определение четвертей матрицы

Используя правила, представленные на рисунке 6, очень легко можно программным путем формировать матрицы требуемого вида.

**Например,** сформировать матрицу N × N вида:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 0 | 0 | 0 | 5 |
| 1  на главной диагонали – 5;  на побочной диагонали – 4;  в I четверти – 0;  во II четверти – 2;  в III четверти – 3;  в IV четверти – 1. | 4 | 0 | 5 | 2 |
| 1 | 1 | 4 | 2 | 2 |
| 1 | 5 | 3 | 4 | 2 |
| 5 | 3 | 3 | 3 | 4 |

Var

mas:array[1..100,1..100] of integer; i,j,n:integer;

Begin

Writeln(‘Введите размерность массива’);

Readln(n);

For i:=1 to n do {заполняем массив в соответствии с правилами}

For j:=1 to n do

Begin

If i=j then mas[i,j]:=4;

If i+j<N+1 then mas[i,j]:=5;

If (i<j) and (i+j<N+1) then mas[i,j]:=0;

If (i<j) and (i+j>N+1) then mas[i,j]:=2;

If (i>j) and (i+j>N+1) then mas[i,j]:=3;

If (i>j) and (i+j<N+1) then mas[i,j]:=4;

End;

For i:=1 to n do {выводим полученный массив на экран в виде таблицы}

Begin

For j:=1 to n do

Write(mas[i,j],’ ‘);

Writeln;

End;

End.

**5. Открытые массивы**

Открытые массивы используются в процедурах и функциях как параметры, у которых не задаются размеры. Фактический размер в этом случае определяется с помощью функций High. Индексация всегда начинается с нуля.

**Например,**

Вычислить максимальный элемент в массиве

function Max (Var Mas: array of integer): integer;

Var Ma : integer;

i : Byte;

Begin

Ma : = Mas [0];

for i : = 1 to High (Mas) do

if Ma < Mas [i] then

Ma : = Mas [i];

Max : = Ma

End.

Данная функция может работать с любым одномерным массивом целых чисел.

**Список литературы**

1. А. Епанешников, В. Епанешников Программирование в среде Turbo Pascal 7.0 - М.: «Диалог-Мифи», 1998
2. Информатика: Учеб. пособие для студ. пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хеннер – М.: Изд. Центр «Академия», 2001
3. Мизрохи. Turbo Pascal и объектно-ориентированное программирование. – М.: Финансы и статистика, 1992
4. Программирование в Delphi – М.: ЗАО «Издательство БИНОМ», 2000
5. Турбо Паскаль 7.0. Начальный курс. Учебное пособие. – М.: «Нолидж», 1998