Агеева Алиса М80-110Б-21

Вариант 2

08.04.22

Теория 2

1. **5 вопрос**: Массивы — упорядоченная совокупность нескольких однотипных объектов (чисел, символов, строк и т. п.), рассматриваемых как единое целое.

Структура описания массива <имя>: ARRAY [нач. номер\_элемента. . конечный\_номер\_элемента] OF <тип>; Например: var a: array [1. . 10] of Real; b: array [0. . 50] of Char; c: array [0. . 4] of byte; d: array [-5. . 10] of integer; Здесь ARRAY, OF — ключевые слова (массив, из). За словом ARRAY в квадратных скобках указывается тип-диапазон, определяющий общее количество элементов в массиве.

Описание многомерных массивов • Способ 1: type mas 1=array [1. . 10] of array [1. . 15] of real; var arr: mas 1; • Способ 2 (упрощенный): type mas 2=array [1. . 10, 1. . 15] of real; var arr: mas 2; • Способ 3 (непосредственный) var mas: array [1. . 10, 1. . 15] of real;

Обращение к элементу многомерного массива • Чтобы обратиться к элементу многомерного массива, нужно указать имя переменной массива и в квадратных скобках перечислить индексы: • a[3, 6]

Ввод и вывод массива • Вводить и выводить массивы можно только поэлементно. Program massiv\_1; // рассмотрим другой вариант описания массива const n = 5; type mas = array[1. . n] of integer; var a: mas; // в данном случае можно было не писать все предыдущие строки, //а записать в разделе var просто a: array[1. . n] of integer; i: byte; begin writeln('введите элементы массива'); for i: =1 to n do readln(a[i]); writeln('вывод элементов массива: '); for i: =1 to n do writeln(a[i]); end.

Ввод и вывод многомерных массивов • Работа с многомерными массивами почти всегда связана с организацией вложенных циклов. Так, чтобы заполнить двумерный массив (матрицу) случайными числами, используют конструкцию вида: for i: =1 to m do for j: =1 to n do a[i, j]: =random(10); • Для вывода двухмерной матрицы на экран можно использовать такой цикл: for i: =1 to m do for j: =1 to n do writeln(a[i, j]); end;

Обработка массивов

При скалярной обработке данных командное слово *i* считывается из памяти команд (MI) в регистр команд скалярного процессора. В адресном поле Аj командного слова содержится указание на данные *j* (адрес данных в памяти данных (MD)). Одновременно со считыванием данных *j* из MD, считываются данные *k* из регистра, заданного в поле R командного слова *i*. Над данными *j* и *k* в арифметико-логическом устройстве (АЛУ) выполняется операция, задаваемая полем «код операции» (КОП) командного слова. Результаты операции заносятся в регистр R.

Обработать массив могут циклы (for, while, until … ) а так же встроенные команды min, max, sort и т.д а так же Передача массива в функцию

Функция может получать на вход массив. В действительности в функцию передается адрес начала массива и его длина. Прототип функции может быть оформлен либо так:

int print\_array(double x[], int len);

1. **8 вопрос**

Интерпретатор командного языка (ИКЯ) получает запросы на выполнение команд через обращения к ОС (с использованием прерывания). Эти команды определяют обращения к функциям. Часть этих функций находится в системной области оперативной памяти. Другая часть – внешние, они размещаются на жёстком диске и вызываются на использование в транзитную часть системной области.

**Регулярные выражения UNIX**

* Звездочка -- \* --

Означает любое количество символа в строке, предшествующего “звездочке”, в том числе и нулевое число символов.

* Точка -- . --

Означает не менее одного любого символа

* Символ -- ^ --

Означает начало строки, но иногда, в зависимости от контекста, означает отрицание в регулярных выражениях.

* Знак доллара -- $ --

В конце регулярного выражения соответствует концу строки.

* Обратный слеш -- \ --

Служит для экранирования специальных символов, это означает, что экранированные символы должны интерпретироваться буквально, т.е. как простые символы (в некоторых случаях наоборот).

* Экранированные "угловые скобки" -- \<...\> --

Отмечают границы

Подстановка команд -- это подстановка результатов выполнения команды [1] или даже серии команд; буквально, эта операция позволяет вызвать команду в другом окружении.

Классический пример подстановки команд -- использование обратных одиночных кавычек (`...`). Команды внутри этих кавычек представляют собой текст командной строки.

name=`basename $0`

echo "Имя этого файла: $ name."

Подстановка переменных

Можно подставлять переменную со знаком $

Пример:

var=Pogoda

echo “$var”

Pogoda

Подстановка команд позволяет даже записывать в переменные содержимое *целых файлов*, с помощью перенаправления или команды cat.

variable1=`<file1` # Записать в переменную "variable1" содержимое файла "file1".

variable2=`cat file2` # Записать в переменную "variable2" содержимое файла "file2"

Встроенный документ (here document) является специальной формой перенаправления ввода/вывода, которая позволяет передать список команд интерактивной программе или команде, например ftp, telnet или ex. Конец встроенного документа выделяется "строкой-ограничителем", которая задается с помощью специальной последовательности символов <<. Эта последовательность -- есть перенаправление вывода из файла в программу, напоминает конструкцию interactive-program < command-file, где command-file содержит строки:

command #1

command #2