1)Массив представляет собой структуру данных, позволяющую хранить совокупность значений любого типа. Массив характеризуется своим именем, типом хранимых элементов, размером (числом хранимых элементов), нумерацией элементов и размерностью. Массивы могут быть статическими и динамическими, одномерными и многомерными.

2) Размерность массива — это количество индексов, необходимое для однозначной адресации элемента в рамках массива. ... Особенностью массива как структуры данных (в отличие, например, от связного списка) является константная вычислительная сложность доступа к элементу массива по индексу.

3) Статическим массивом называется такой массив, число элементов в котором постоянно и заранее определено и задается программистом при разработке программы.

4) Размерность — это количество элементов в массиве.

Размерность массива и тип его элементов определяют объем памяти, который необходим для хранения массива, поэтому размерность — это целое положительное константное выражение.

5) Перед использованием массива его необходимо инициализировать, т. е. присвоить элементам массива начальные значения. Начальные значения элементам массива можно задавать при объявлении массива Например: int A[10] ={2,4,3,6,3,6,1,23,5,25};// массив целых чисел; char S[10] ={“abcdefghi\0”};// массив символов По принятому соглашению массивы символов, содержащие строку, после конца строки обязательно должны содержать нулевой символ. По этой причине в массиве символов S[10] присутствуют только девять значащих символов, и последний элемент является нулевым символом. Если начальных значений меньше, чем элементов массива, то оставшиеся элементы автоматически инициализируются. Это позволяет инициализировать символьный массив в более удобной форме: char kaf\_name [ ] = ”кафедра ВТиИС”; При объявлении со списком инициализации размер массива можно не указывать, тогда количество элементов массива будет равно количеству элементов в списке начальных значений. Например, объявление Int A [] = {1,2,3,4,5}; создает массив из пяти элементов. Если многомерный массив инициализируется при объявлении, список значений по каждой размерности заключается в фигурные скобки. Например, инициализация при объявлении трехмерного массива записывается в виде: Int A3[4] [3] [2] = { {{0,1}, {2,3}, {4,5}}, {{6,7}, {8,9}, {3,8}}, {{1.3}, {5,2}, {8,5}}, {{7,3,1}, {4,5}, {2,9}} }; Эта запись создает массив А3, четыре строки которого являются матрицами размерностью

6) Многомерные массивы в C++ рассматриваются как массивы, элементами которых являются массивы.

Определение многомерного массива должно содержать информацию о типе, размерности и количестве элементов каждой размерности. Многомерный массив подобно одномерному массиву может быть проинициализирован с помощью списка инициализаторов.

7) Если многомерный массив инициализируется при объявлении, список

значений по каждой размерности заключается в фигурные скобки. Например,

инициализация при объявлении трехмерного массива записывается в виде:

Int A3[4] [3] [2] = { {{0,1}, {2,3}, {4,5}}, {{6,7}, {8,9}, {3,8}},

{{1.3}, {5,2}, {8,5}}, {{7,3,1}, {4,5}, {2,9}} };

Эта запись создает массив А3, четыре строки которого являются матрицами размерностью (3 х 2).

8) При статической (определяемой на этапе компиляции) инициализации значения C-массива перечисляются в порядке указания размеров (индексов) в определении массива. Каждый уровень (индекс), кроме самого младшего, многомерного массива заключается в свою пару фигурных скобок. Значения самого младшего индекса указываются через запятую:

const unsigned int DIM1 = 3;

const unsigned int DIM2 = 5;

int ary[DIM1][DIM2] = {

{ 1, 2, 3, 4, 5 },

{ 2, 4, 6, 8, 10 },

{ 3, 6, 9, 12, 15 }

};

В примере показана статическая инициализация прямоугольного массива. Весь список инициализирующих значений заключён в фигурные скобки. Значения для каждой из 3 строк заключены в свою пару из фигурных скобок, значения для каждого из 5 столбцов для каждой строки перечислены через запятую.

При наличии инициализатора, самый левый размер массива может быть опущен. В этом случае компилятор сам определит этот размер, исходя из списка инициализации.

9) При вызове функции указывается просто имя массива. Например, обращение к такой функции может быть записано в виде: const Amax =10; int Bs[Amax]; F(Bs); В большинстве случаев только имени массива мало, чтобы провести в функции обработку его элементов, Внутри функции требуется знать размер массива, чтобы можно было организовать его циклическую обработку, поэтому в функцию передается не только массив, но и его размер.

10) Динамическое распределение памяти в этой области может производиться несколькими способами: с помощью операций new и delete или с помощью библиотечных функций malloc, calloc, realloc, free. Указанные функции объявлены в файле stdlib.h или alloc.h. Распределение памяти с помощью операций new и delete. Следующий фрагмент использует оператор new для получения указателя на N-байтный массив, число элементов которого N вводится пользователем: int N N = StrToInt (Edit1->Text) int \*Barray = new int[N]; После выделения памяти под динамический массив необходимо проверить, успешно ли завершилась эта операция. Для этого можно использовать следующую конструкцию. if (Barray != NULL) ShowMessage ("Память успешно выделена") else { ShowMessage ( " Ошибка выделения памяти "); exit (1); } Программа сразу проверяет значение, присвоенное оператором new переменной-указателю. Если указатель содержит значение NULL, значит new не смог выделить запрашиваемый объем памяти. Если же указатель содержит не NULL, следовательно, new успешно выделил память и указатель содержит адрес начала блока памяти. Если new не может удовлетворить запрос на память, он возвратит NULL. Для размещения в памяти двумерного массива необходимо вначале задать количество строк и столбцов. Затем создать массив указателей на массив строк и потом циклом задать число элементов в каждой строке. Например: int N = StrToInt( Edit1-Text); // число строк int M = StrToInt( Edit1-Text);// число столбцов int\*\* Array = new int \*[N]; // массив указателей на массив строк for ( int I = 0; iText); Carray = (int\*) malloc(N); …………………. Free(Carray); В данном примере функция malloc выделяет память под массив из N целых чисел, где значение N вводится пользователем. После выполнения с массивом каких-либо операций память освобождается функцией free.

11) Для того, чтобы изменить число элементов в массиве созданного оператором new, необходимо создать вспомогательный массив с числом элементов, равным исходному. Затем переместить элементы из исходного массива во вспомогательный. Увеличить размер исходного массива и переместить в него элементы из вспомогательного массива и удалить вспомогательный массив. Распределение памяти с помощью функций malocc, calloc, relloc. Для того, чтобы указанные функции были объявлены в файле stdlib.h alloc.h, которые записываются в заголовочный файл формы, функция malloc выделяет блок памяти размером size байтов. В случае успешного выделения памяти возвращает указатель на выделенный блок, в противном случае возвращается NULL. Объявление функции следующее: void \*malloc (size\_t size); Функция calloc выделяет память под несколько объектов, размер каждого из которых равен size. Функция объявляется следующим образом: void \*calloc (size\_t n\_object, size\_t size); Общий объем выделенной памяти для n объектов составляет n\_object\* size. В случае успешного выделения памяти возвращает указатель на выделенный блок, в противном случае возвращается NULL. Функция realloc позволяет изменить размер ранее выделенного блока памяти. Функция объявляется следующим образом: void \*realloc (void \*name\_block, size\_t size); Данная функция изменяет размер блока, на который указывает name\_block, до размеров size. При этом предполагается, что block указывает блок памяти, выделенный ранее функциями malloc, calloc или realloc. Если блок нового размера не может быть выделен, то функция realloc возвращает NULL. Если же память выделилась успешно, то возвращается адрес выделенного блока. При этом он может отличаться от начального значения name\_block, поскольку функция при необходимости осуществляет копирование содержимого блока в новое место. Если размер size задан равным нулю, то выделенный ранее блок, на который указывает name\_block, освобождается, а функция возвращает NULL. Функция free объявляется следующим образом: void free(void \*block); Она освобождает блок памяти, на который указывает block, выделенный ранее функциями malloc, calloc или realloc. Примеры использования описанных функций: 1) int \*Carray; int N; N = StrToInt(Edit1->Text); Carray = (int\*) malloc(N); …………………. Free(Carray); В данном примере функция malloc выделяет память под массив из N целых чисел, где значение N вводится пользователем. После выполнения с массивом каких-либо операций память освобождается функцией free. 2) В этом примере можно было использовать для выделения памяти функцию calloc; Carray = (int\*) calloc (N) , sizeof(int)); 3) Для изменения выделенной памяти применим функцию realloc. Carray = (int\*) realloc ( Carray, N1); Или более простой оператор: realloc(Carray, N1);

12)

13) Вначале объявите массив с заданным числом элементов. Для генерации элементов массива используйте датчик случайных, для инициализации массива – цикл со счетчиком. Для вывода созданного массива используйте объект ListBox. Для поиска максимального элемента, вычисления суммы элементов и среднего значения элемента массива – цикл со счетчиком, в теле которого разместите выражение для поиска максимального элемента, выражение для суммирования членов массива, после цикла – выражение для вычисления среднего значения элемента массива.

14) Для вывода значений элементов матрицы – объект StringGrid, основными используемыми свойствами которого являются: RowCount – количество строк; ColCount – количество столбцов; FixedRows – количество фиксированных строк; FixedCols – количество фиксированных столбцов.

15)

16)

17)Метод Cells[i][j] = Array[i][j] – заполнение ячеек таблицы элементами массива Array программным способом. Для того чтобы можно было заполнять таблицу StringGrid с клавиатуры, необходимо установить в свойстве Options значение goEditing, равным True.

18) ColCount – количество столбцов;

RowCount – количество строк;

18)Удаление памяти осуществляется операцией delete. Например: delete Barray; Barray = NULL; Для того, чтобы изменить число элементов в массиве созданного оператором new, необходимо создать вспомогательный массив с числом элементов, равным исходному. Затем переместить элементы из исходного массива во вспомогательный. Увеличить размер исходного массива и переместить в него элементы из вспомогательного массива и удалить вспомогательный массив