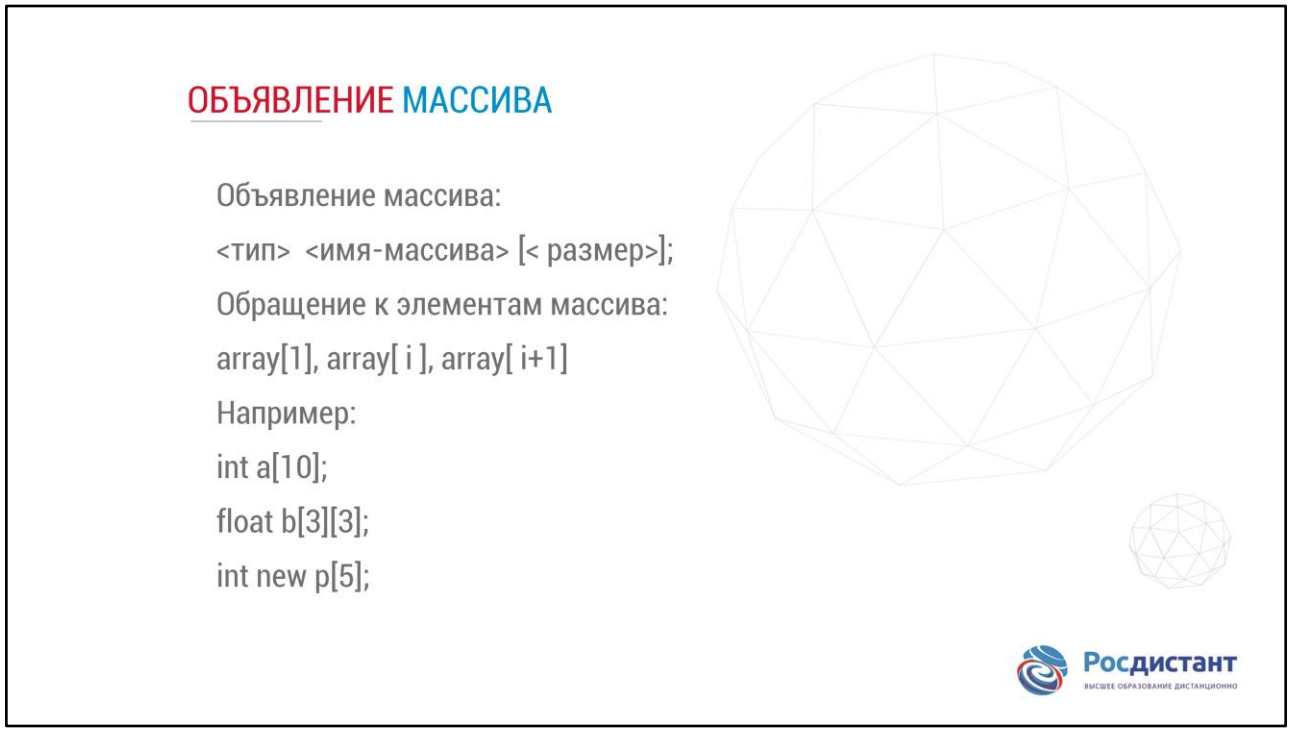

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО
ОСНОВЫ
ПРОГРАММИРОВАНИЯ
5 ЧАСТЬ



Слайд 103

Тема 5.1 Массивы

Обработка большого объема информации приводит к необходимости использования массивов.

Массив – это поименованный набор однотипной информации.

Массив – это группа элементов одинакового типа. В дальнейшем мы рассмотрим возможности хранения информации разного типа в одном массиве с помощью пользовательских типов данных, структур.

«Напомним, что массив – это упорядоченный набор величин, обозначаемых одним именем. Данные, являющиеся элементами массива, располагаются в памяти компьютера в определенном порядке, который задается индексами, порядковыми номерами элементов массива. В **C++** массив, как и любая переменная, должен быть объявлен. Делается это с помощью служебного слова, указывающего тип. Смешанные массивы не допустимы.»

Имя массива - уникальный идентификатор, строится по тем же правилам, что и имена простых переменных. Тип элементов массива определяет количество байт, необходимое для хранения каждого элемента.

Согласно объявлению массива выделяется память под хранение массива. Имя массива, это фактически, указатель, который хранит адрес начала области

памяти, выделенной под массив.

Весь набор информации объединен общим именем, а доступ к отдельным элементам массива осуществляется с помощью индекса. Индекс указывается рядом с именем массива в квадратных скобках. Индексом может быть числовая константа, переменная или выражение целого типа.

«При работе с массивами необходимо внимательно следить за тем, чтобы не выходить за их объявленные границы. Компилятор **C++** , в отличие, например, от Паскаля, не предупреждает об этой ошибке! Попытка ввести больше элементов, чем описано, приведет к неверным результатам, а попытка вывести – выведет случайный результат, находящийся в памяти.».

ИНИЦИАЛИЗАЦИЯ ЭЛЕМЕНТОВ ОДНОМЕРНЫХ МАССИВОВ

Определение массива	Результаты инициализации
int a[5] = {5,10, 20};	a[0]=5, a[1]=10, a[2]=20, a[3]=0, a[4]=0
char b[ ]={'A','B','C', 'D'};	b[0]='A', b[1]='B', b[2]='C', b[3]='D'
char c[ ] = "ABCD";	c[0]='A', c[1]='B', c[2]='C', c[3]='D', c[4]='\0'
sizeof (< имя- массива >) / sizeof (< имя-массива >[0])	

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 104

На слайде описаны и инициализированы три массива. Определены начальные значения элементов массива. Индексы массивов начинаются с нуля, то есть в массиве всегда присутствует нулевой элемент.

Массив **a** – это массив, целых чисел, состоящий из пяти элементов. Значения первых четырех элементов этого массива определены, элементы четвертый и пятый – получат нулевые значения.

Массив **b** – это символьный массив, его размер в явном виде не задан, о чем свидетельствуют пустые квадратные скобки рядом с именем массива. Размер будет определен количеством элементов при начальной инициализации, то есть массив **b** – состоит из четырех элементов.

Массив **c** – также массив символов. При определении элементов массива, как показано в третьем примере, в конце цепочки символов автоматически ставится признак конца строки символов - **слеш ноль**. То есть в массиве **c** пять элементов.

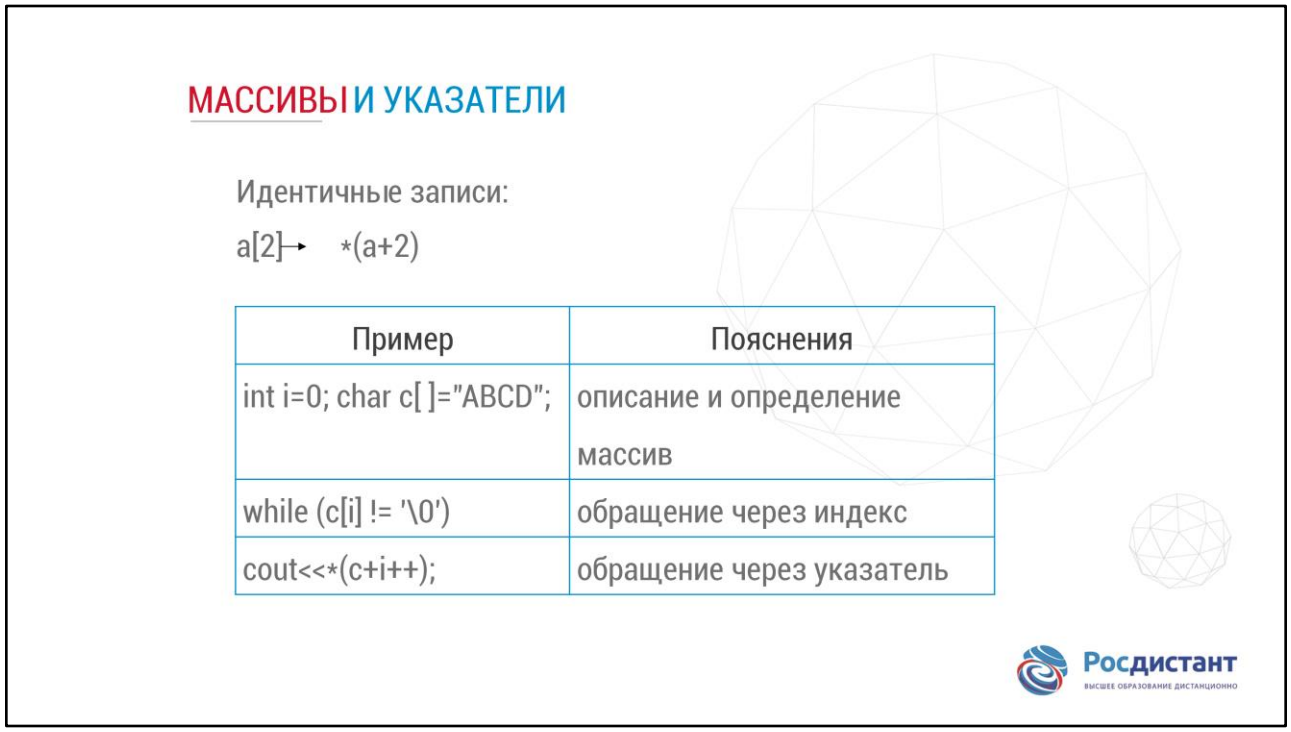
Если размер массива при описании отсутствует, список начальных значений массива в определении массива обязателен.

С помощью операции **sizeof** можно определить размер массива. В формуле:

- делимое – имя массива, определяет размер всего массива в байтах;

- делитель – элемент массива, например, первый, определяет размер в байтах одного элемента массива.

По этой формуле определяется размер массива, то есть количество элементов в массиве.



Слайд 105

Как говорилось выше, при обращении к элементам массива используют индексы. Индекс определяет не номер элемента, а его смещение относительно начала массива. Имя массива с индексом в квадратных скобках фактически является выражением с двумя операндами:

имя массива – это первый операнд, адрес начала массива в памяти;

индекс – второй операнд, выражение целого типа, определяющее смещение от начала массива. Смещение, то есть на какое количество байт, относительно начала массива - определяется типом элементов массива.

Используя операцию – звездочка, обращение по адресу, на слайде продемонстрированы две записи абсолютно идентичные.

Приведены пример, демонстрирующий два варианта обращения к массиву, через индекс и через указатель.

Во второй строке таблицы продемонстрировано обращение к элементам массива, используются индексы. Цикл продолжается, пока не встретится символ - признак конца строки.

В третьей строке таблицы - обращение к элементам массива происходит через указатель. Имя массива и есть указатель на начало массива. А индекс – смещение относительно начала массива.

ВВОД-ВЫВОД ЭЛЕМЕНТОВ МАССИВА

№	Пример
1	int int_arr[] = {0,1,2,3,4,5,6}; for (int i=0; i<7; i++) cout<<int_arr[i];
2	int int_arr[] = {0,1,2,3,4,5,6}; foreach(int numder, int_arr) { std::cout << number ; }

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 106

На слайде приведены два примера вывода на экран значений элементов массива.

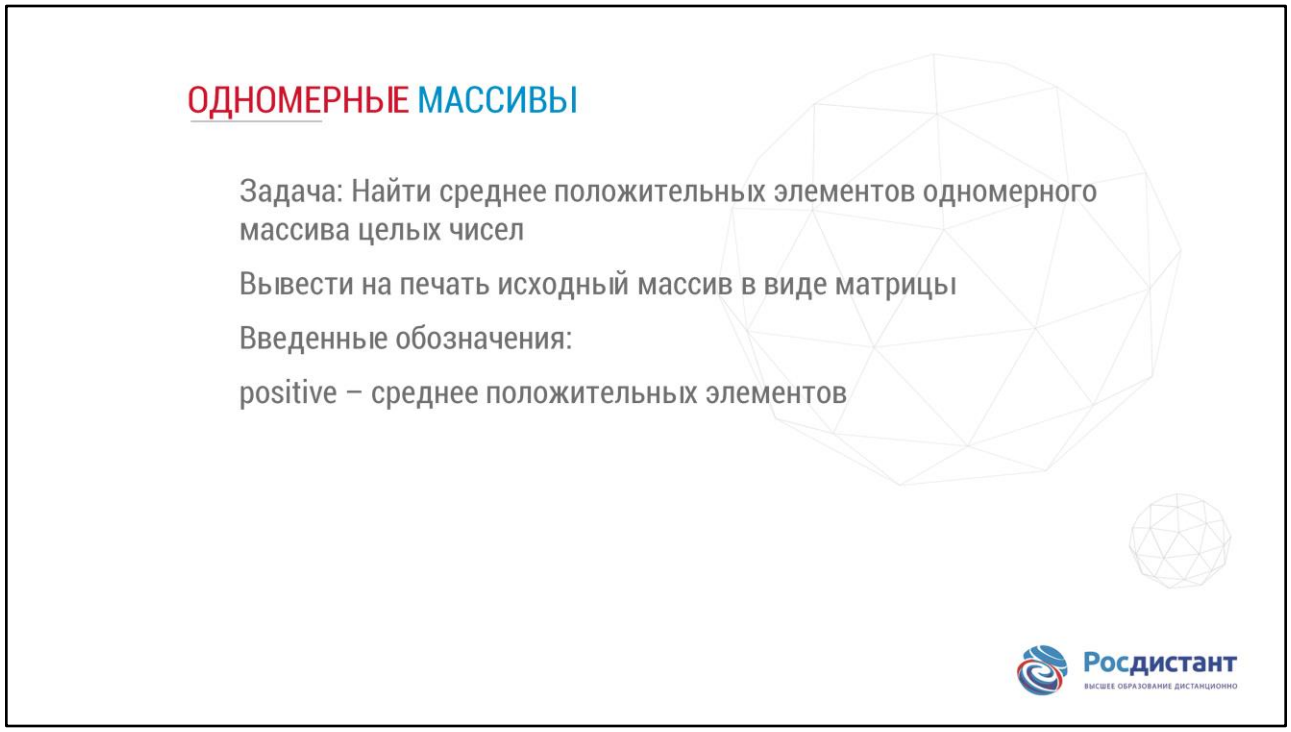
В первом примере, описан массив и определены его значения. Вывод элементов осуществляется в цикле, в котором перебор элементов осуществляется с помощью индекса массива.

Во втором примере использован оператор цикла **foreach .**

Новый тип цикла **— foreach,** цикл, основанный на диапазоне**,** предоставляет более простой и безопасный способ итерации по массиву.

При выполнении цикла переменной **number** присваивается значение первого элемента, т.е. значение ноль. Дальше программа выполняет вывод значения переменной **number** Затем цикл выполняется снова, и значением **number** уже является второй элемент массива. Вывод значения **number** выполняется снова. Цикл продолжает своё выполнение до тех пор, пока в массиве не останется не пройденных элементов.

Обратите внимание, переменная **number** не является индексом массива. Ей просто присваивается значение элемента массива в текущей итерации цикла.



Слайд 107

Рассмотрим примеры обработки массивов. На слайде – формулировка задачи.

В условии задачи указано, что массив целых чисел.

Для нахождения среднего значения элементов массива введена новая переменная **positive**, которая должна иметь вещественный тип. Данная переменная описана как вещественная, чтобы при подсчете среднего значения элементов массива, то есть при делении на количество элементов не исказился результат. Известно, что при делении целочисленных переменных, дробная часть теряется.

Переменные, описанные в программе, автоматически не обнуляются. Поэтому необходимо предварительно обнулить значение переменной **positive.**

В данной задаче реализуется ввод размера массива и значений элементов массива с клавиатуры.

Ввод осуществляется в цикле, причем, каждое прохождение через цикл соответствует вводу одного элемента массива.

Если в массиве отсутствуют положительные числа, вывести на печать соответствующее сообщение.

ОДНОМЕРНЫЕ МАССИВЫ

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{ int n;
  cout<<"\n Введите n  ";
  cin>>n;    //ввод размера массива
  int array[n]; float positive=0;
```


Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

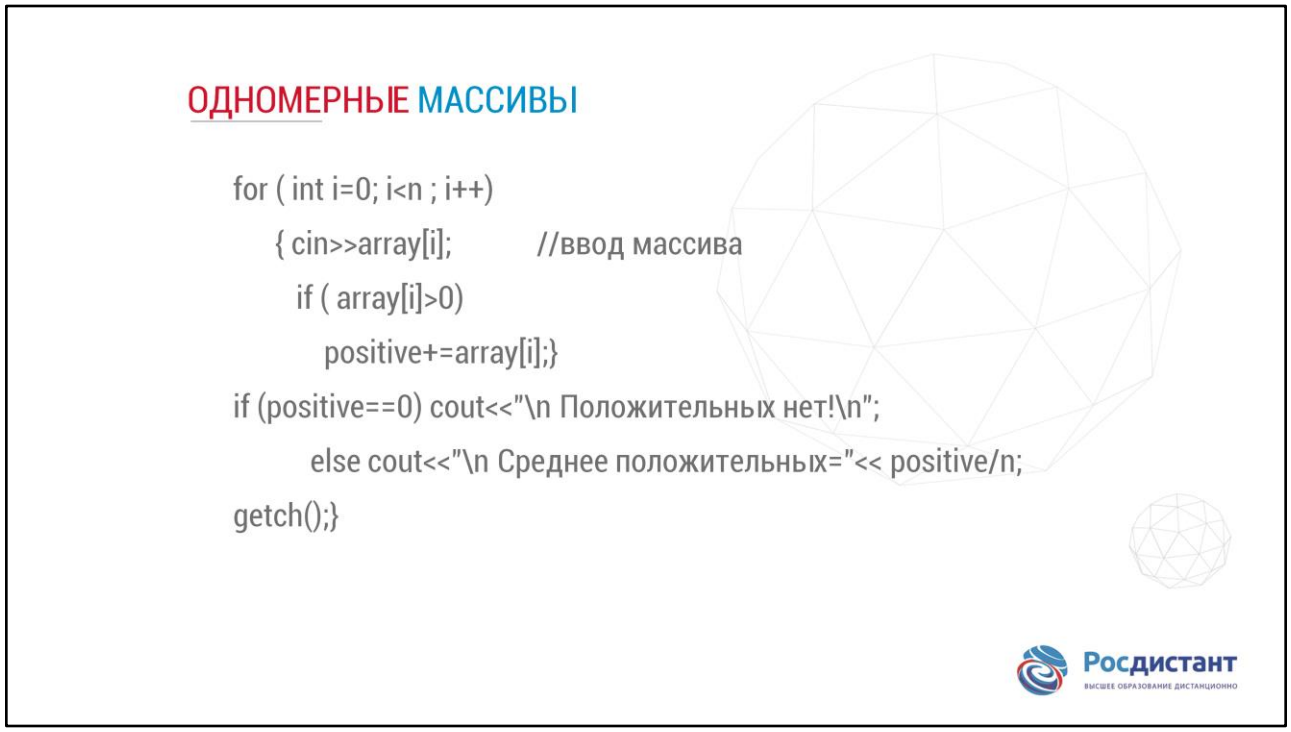
Слайд 108

На слайде первый фрагмент программы. С клавиатуры вводится конкретный размер массива **n .** Описан массив, элементами которого являются целые числа. Размер массива заранее не определяется, чтобы избежать избыточного выделения памяти под хранение элементов массива.

Идентификатор **array** – имя массива.

В отличии от некоторых алгоритмических языков, на языке **C++** значения переменных, описанных в программе, автоматически не обнуляются. Поэтому, переменную **positive** , в которой будет подсчитываться среднее значение положительных элементов массива, необходимо обнулить.

Об этом факте не следует забывать, чтобы не произошло искажения результатов вычислений.



Слайд 109

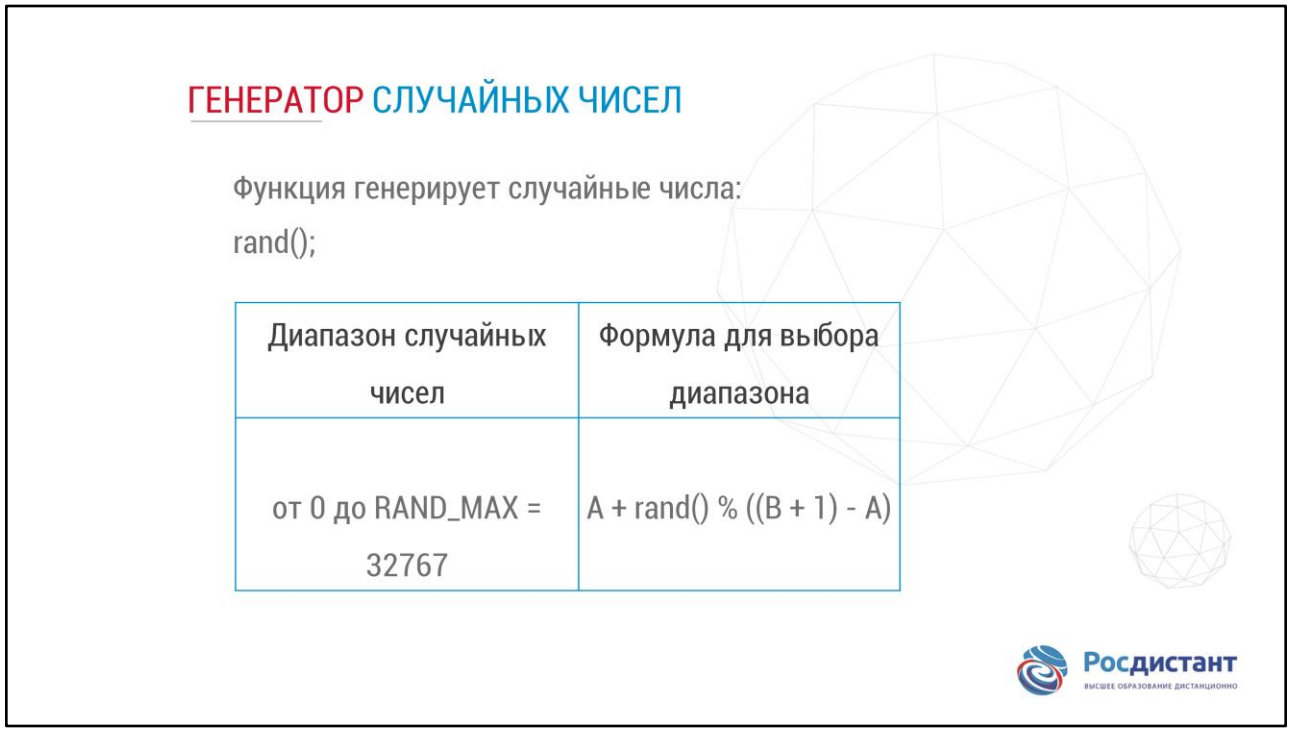
На слайде демонстрируется продолжение программы.

Обратите внимание, в одном цикле предусмотрен ввод элементов массива и подсчет суммы положительных элементов массива. Используется оператор цикла с параметром. Последовательность операторов, образующих тело цикла, необходимо заключить в фигурные скобки.

Параметр цикла, переменная **i ,** описана в структуре оператора цикла, то есть тело цикла является блоком. Об этом мы уже говорили ранее.

И не следует забывать, что переменная, описанный внутри блока, не видна вне этого блока, и время ее существования – от начала и до конца работы данного блока.

После выполнения цикла, анализируется значение переменной **positive**, и, если оно равно нулю, выдается сообщение об отсутствии в массиве положительных чисел. Данное действие реализует свойство алгоритма – результативность, то есть или программа печатает результат, или сообщение о причинах отсутствия результата.



Слайд 110

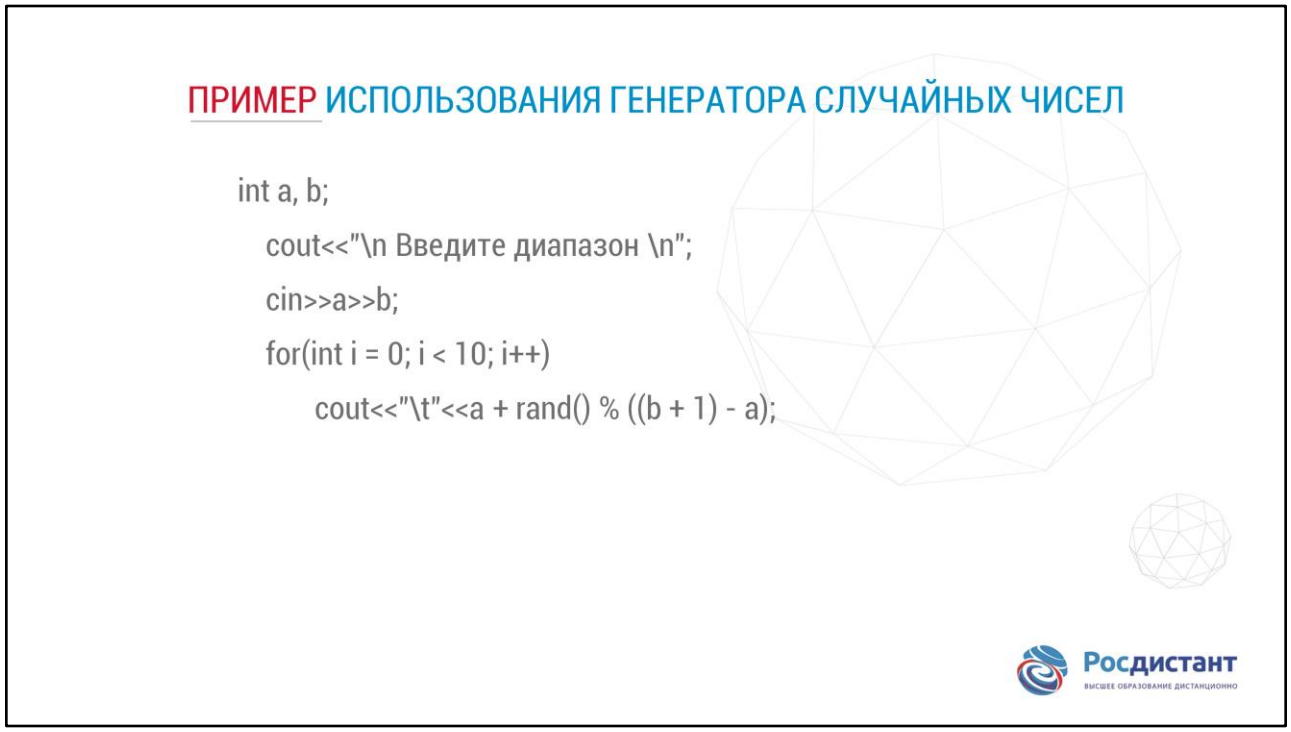
Возможность генерировать случайные числа очень полезна в некоторых задачах, в частности в программах научного или статистического моделирования.

Компьютеры неспособны генерировать случайные числа. Вместо этого они могут имитировать случайность, что достигается с помощью генераторов псевдослучайных чисел.

Функция **rand** генерирует случайное число в последовательности.

В большинстве случаев нам не нужны рандомные числа между 0 и максимально возможным числам - нам нужны числа между двумя другими значениями. Например, если нам нужно сымитировать бросок кубика, то диапазон значений будет невелик: от 1 до 6.

Чтобы сгенерировать числа в диапазоне от A до B включительно, можно использовать формулу, указанную таблице.



Слайд 111

На слайде продемонстрирован фрагмент программы, использующий генератор случайных чисел.

Задавая значения предельных значений, минимального и максимального, можно сгенерировать определенное количество случайных чисел из указанного диапазона.

Генератор случайных чисел – это программа, которая принимает стартовое, начальное значение и выполняет с ним определённые математические операции, чтобы конвертировать его в другое число, которое совсем не связанное со стартовым. Затем программа использует новое сгенерированное значение и выполняет с ним те же математические операции, что и с начальным числом, чтобы конвертировать его в ещё в одно новое число – третье. Применяя этот алгоритм к последнему сгенерированному значению, программа может генерировать целый ряд новых чисел, которые будут казаться случайными

Если вы запустите программу несколько раз, то заметите, что в результатах всегда находятся одни и те же числа! Это означает, что, хотя каждое число в последовательности кажется случайным относительно предыдущего, вся последовательность не является случайной вообще! А это, в свою очередь, означает, что наша программа полностью предсказуема. Одни и те же значения ввода приводят к одним и тем же значениям вывода. Бывают случаи, когда это

может быть полезно или даже желательно.

ПРИМЕР ИСПОЛЬЗОВАНИЯ ГЕНЕРАТОРА СЛУЧАЙНЫХ ЧИСЕЛ

Задача: найти наибольший среди отрицательных элементов одномерного массива целых чисел. Определить номер максимального элемента массива

Введенные обозначения	Пояснения
max_array	максимальный элемент
number	номер максимального элемента
flag	признак наличия отрицательных элементов, флажок

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 112

Рассмотрим задачу поиска максимального элемента в одномерном массиве. Дополнительным является условие – среди отрицательных элементов.

При решении подобных задач необходимо обратить внимание, какое начальное значение необходимо присвоить переменной, которой присваивается значение максимального элемента. Если бы не было дополнительного условия, поиск максимального среди отрицательные элементов массива, достаточно этой переменной присвоить первоначально значение любого из элементов массива, например, первого элемента.

В нашей задаче, этой переменной необходимо присвоить значение элемента массива из множества отрицательных элементов массива. Это важно!

На слайде приведена таблица введенных переменных. Возможен вариант, когда в массиве отрицательные элементы отсутствуют. Именно по этой причине введена переменная **flag**, которая является признаком наличия или отсутствия в массиве отрицательных элементов.

ПОИСК МАКСИМАЛЬНОГО ЗНАЧЕНИЯ

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{  int n, max_array, number, flag=0;
   int array[100];
   cout<<"\n Введите n \n";
   cin>>n;
```

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 113

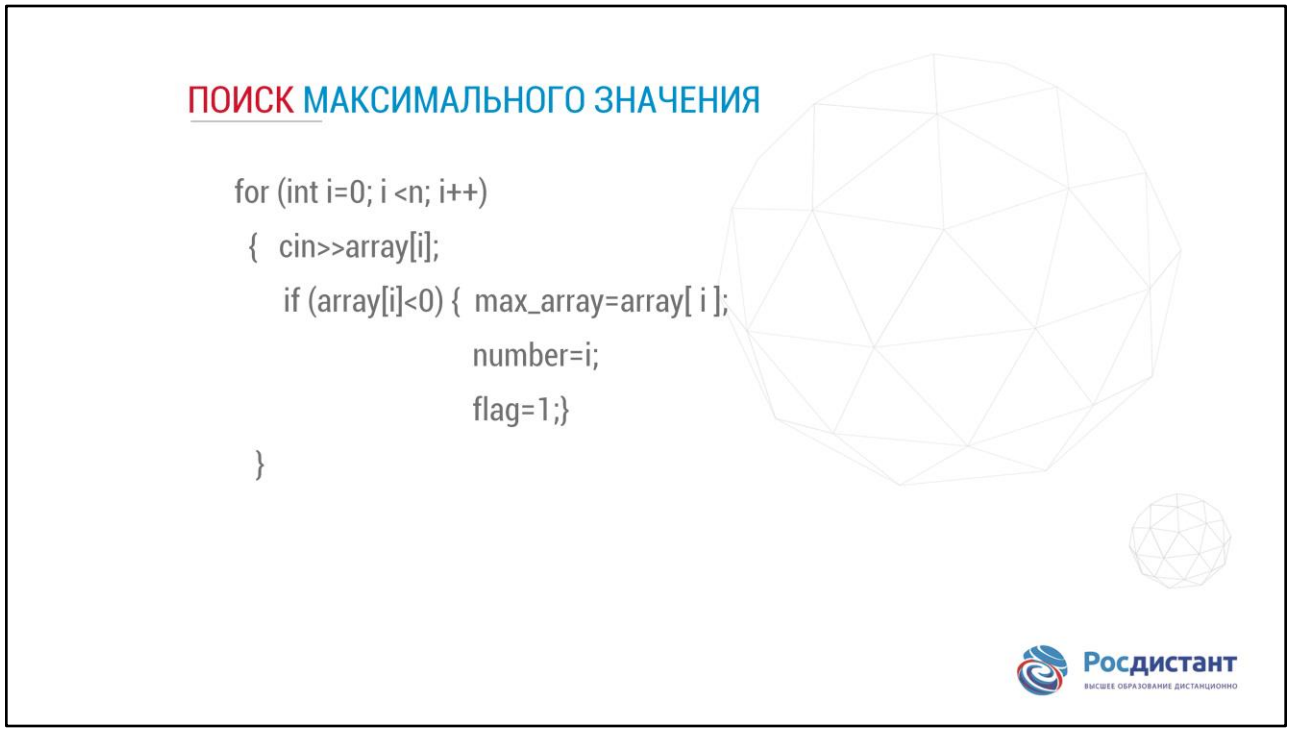
Переменной, отвечающей за максимальный элемент, должна иметь тот же тип, что и элементы массива.

Переменной **flag** , которая является признаком присутствия в массиве отрицательных чисел, первоначально присвоено значение равное нулю.

После перебора элементов массива, если значение переменной **flag** осталось равным нулю это означает, что в массиве отрицательные элементы отсутствуют.

В данной программе описан массив на сто элементов. С клавиатуры вводится количество элементов массива, и это число не может превышать числа сто. Ввод конкретного количества элементов **n** позволяет отладить программу на меньшем количестве элементов.

Если появляется необходимость обработки большего количества элементов, то есть массива большего размера, необходимо внести изменения в программе.



Слайд 114

На слайде - фрагмент ввода элементов массива с клавиатуры. Ввод реализован с помощью оператора цикла с параметром.

В теле цикла присутствует два оператора: ввод элементов и анализ на наличие отрицательных элементов. При вводе отрицательных чисел, в переменную, отвечающую за максимальное значение, присваивается значение отрицательного элемента массива. Не важно, какой это элемент, первый или последний. Главное, что он из множества отрицательных чисел.

При этом переменной **flag** присваивается значение равные единице, свидетельствующий о наличии в массиве отрицательных элементов.

Обратите внимание, номер элемента также сохраняется.

Вспомним о свойстве алгоритма - результативность. То есть, алгоритм должен приводить или к результату, или к сообщению о причинах его отсутствия. Для реализации этого свойства алгоритма и введена переменная **flag.**

ПОИСК МАКСИМАЛЬНОГО ЗНАЧЕНИЯ

```
if ( f lag !=0 )
  {   for (int  i =0;  i<n;  i++)
        if  (array[i]>max_arrya && array[i]<0)
                  { max_array=array[i]; number=i;}
      cout<<"\n array["<<number<<"]="<<max_array; }
      else   cout <<"\n Отрицательных нет ! " ;
getch(); }
```


Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

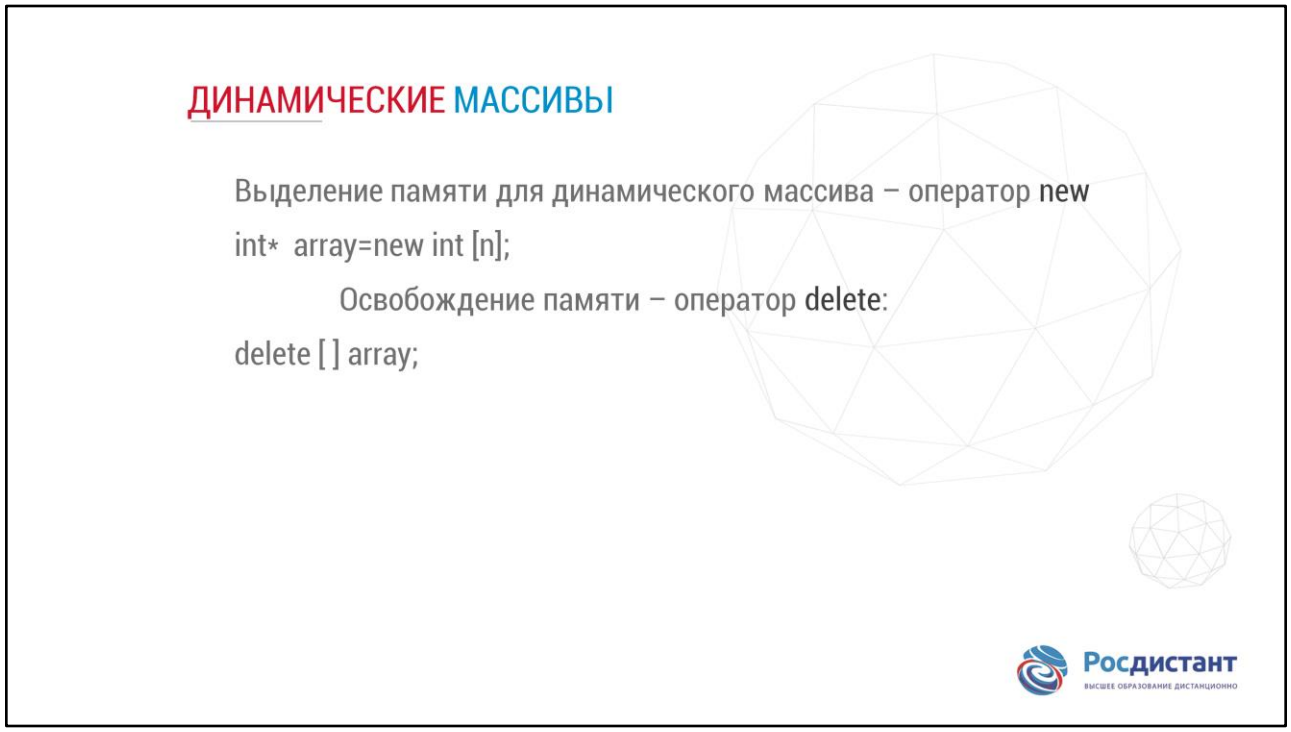
Слайд 115

На слайде - фрагмент поиска максимального среди отрицательных элементов массива.

Анализируется значение переменной **flag,** и, если в массиве есть отрицательные числа, начинается поиск максимального элемента среди отрицательных чисел. Сохраняется номер этого элемента, то есть определяется местоположение максимального отрицательного элемента в общем наборе.

В противном случае выдается сообщение об отсутствии отрицательных чисел в массиве.

Вывод результата предусматривает вывод номера и значение максимального элемента. Обратите внимание, для большей наглядности, при выводе на печать рядом с именем массива в квадратных скобках выведется индекс найденного элемента.



Слайд 116

Указатель - это переменная, значением которой является адрес памяти, ячейка памяти.

Указатели объявляются точно так же, как и обычные переменные, только со звездочкой между типом данных и идентификатором.

Поскольку указатели содержат только адреса, то при присваивании указателю значения — это значение должно быть адресом. Указатели должны иметь тип данных. Без типа указатель не знал бы, как интерпретировать содержимое, на которое он указывает при разыменовании. Об этом мы уже говорили выше.

Память для динамических массивов выделяется из другого резервуара, чем память, используемая для фиксированных массивов, размер массива может быть довольно большим.

Память для большинства обычных переменных, включая фиксированные массивы, выделяется из специального резервуара памяти - **стека**. Объём памяти стека в программе невелик. Если произойдёт переполнение стека, операционная система автоматически завершит выполнение вашей программы.

Эти проблемы легко устраняются с помощью динамического выделения памяти. Динамическое выделение памяти - это способ запроса памяти из операционной системы запущенными программами по надобности. Эта память не выделяется из ограниченной памяти стека программы, а из гораздо большего хранилища,

управляемого операционной системой - **кучи**. На современных компьютерах размер кучи может составлять гигабайты памяти.

Использование динамических массивов очень удобно, когда появляется передача массива в функцию, об этом мы поговорим позднее.

На слайде показано объявление и удаление динамических массивов с помощью операторов **new** и **delete** .

ДИНАМИЧЕСКИЕ МАССИВЫ

Задача: Упорядочить элементы одномерного массива по возрастанию

Введенные обозначения	Пояснения
number	номер максимального элемента массива
flag	признак того, что элементы массива отсортированы, флажок
Функция swap(a,b);	Меняет местами содержимое двух переменных a и b

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 117

«Существует два основных способа хранения информации в оперативной памяти. Первый заключается в использовании глобальных и локальных переменных. В случае глобальных переменных выделяемые под них память остается неизменной на все время выполнения программы.

Под локальные переменные программа отводит память из стекового пространства. Однако локальные переменные требуют предварительного определения объема памяти, выделяемой для каждой ситуации. Хотя язык **C++** эффективно реализует такие переменные, они требуют от программиста заранее знать, какое количество памяти необходимо для каждой ситуации.

Второй способ, которым C++ может хранить информацию, заключается в использовании системы динамического распределения. При этом способе память распределяется для информации из свободной области памяти по мере необходимости. Область свободной памяти находится между кодом программы с ее постоянной областью памяти и стеком. Динамическое размещение удобно, когда неизвестно, сколько элементов данных будет обрабатываться.»

Если до начала разработки программы неизвестно, сколько в массиве элементов, в программе следует использовать динамические массивы. Память под них выделяется с помощью операции **new** в динамической области память.

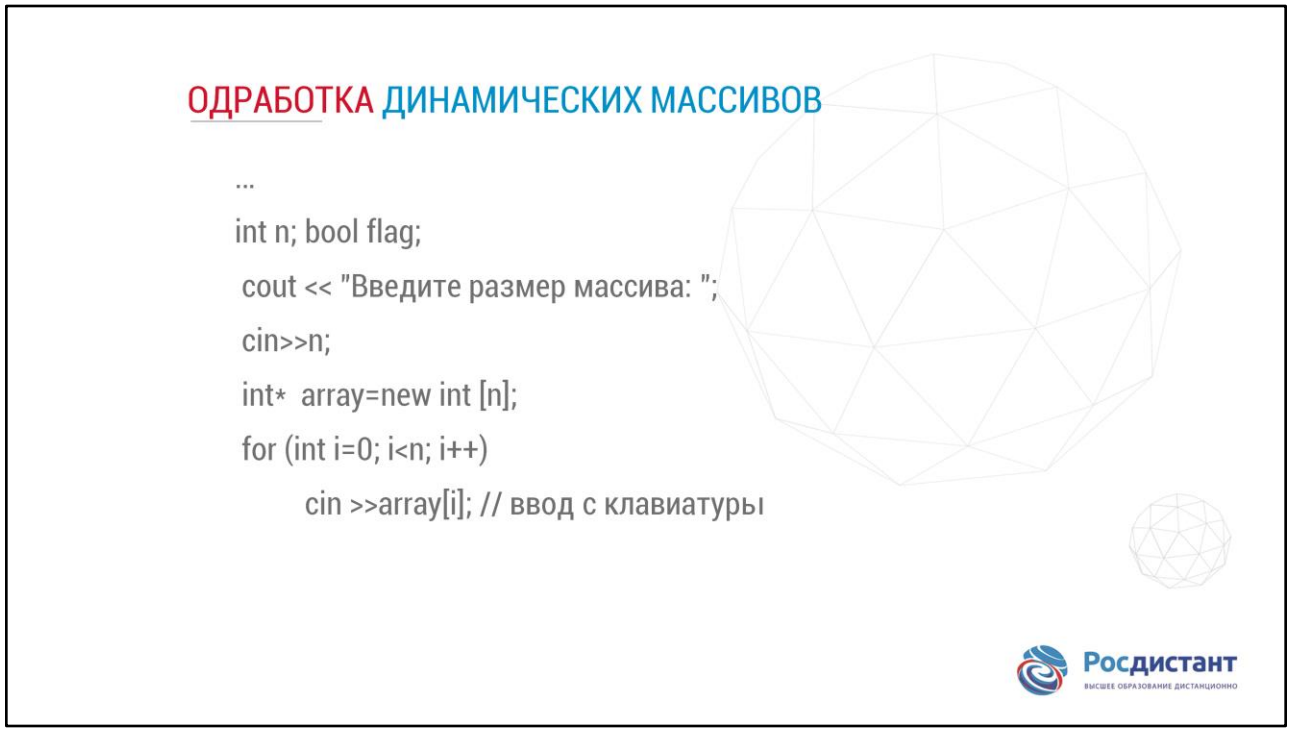
Адрес начала массива хранится в переменной, называемой указателем – имени массива.

В задаче будем использовать динамический массив. Динамическое выделение памяти для массива на языке **C++** позволяет устанавливать его длину во время выполнения программы.

Непосредственно во время выполнения программы можно и освобождать память, отведенную под хранение этого массива. На предыдущем слайде мы рассмотрели эти операторы.

Алгоритм предусматривает сравнение соседних элементов массива, и, если предыдущий элемент окажется больше последующего, эти элементы необходимо поменять местами. Для этого будем использовать функцию функции **swap.**

В последней строке таблицы приведены структура функции **swap ,** которая меняет местами содержимое двух переменных, двух аргументов этой функции. Аргументы указываются рядом с именем функции в скобках и разделяются запятой.



Слайд 118

«Очевидно, что элементы массива, как правило, располагаются в произвольном порядке. Но во многих случаях может понадобиться расположить их, например, в порядке возрастания. Такая процедура упорядочения называется сортировкой.»

Существует различные алгоритмы сортировки. Но эти алгоритмы вы будете подробно изучать в рамках другой дисциплины.

Рассмотрим задачу: отсортировать элементы одномерного массива в порядке возрастания.

Необходимо преобразовать массив таким образом, чтобы каждый предыдущий элемент массива, был меньше последующего элемента.

Переменная **n** – определяет размер массива.

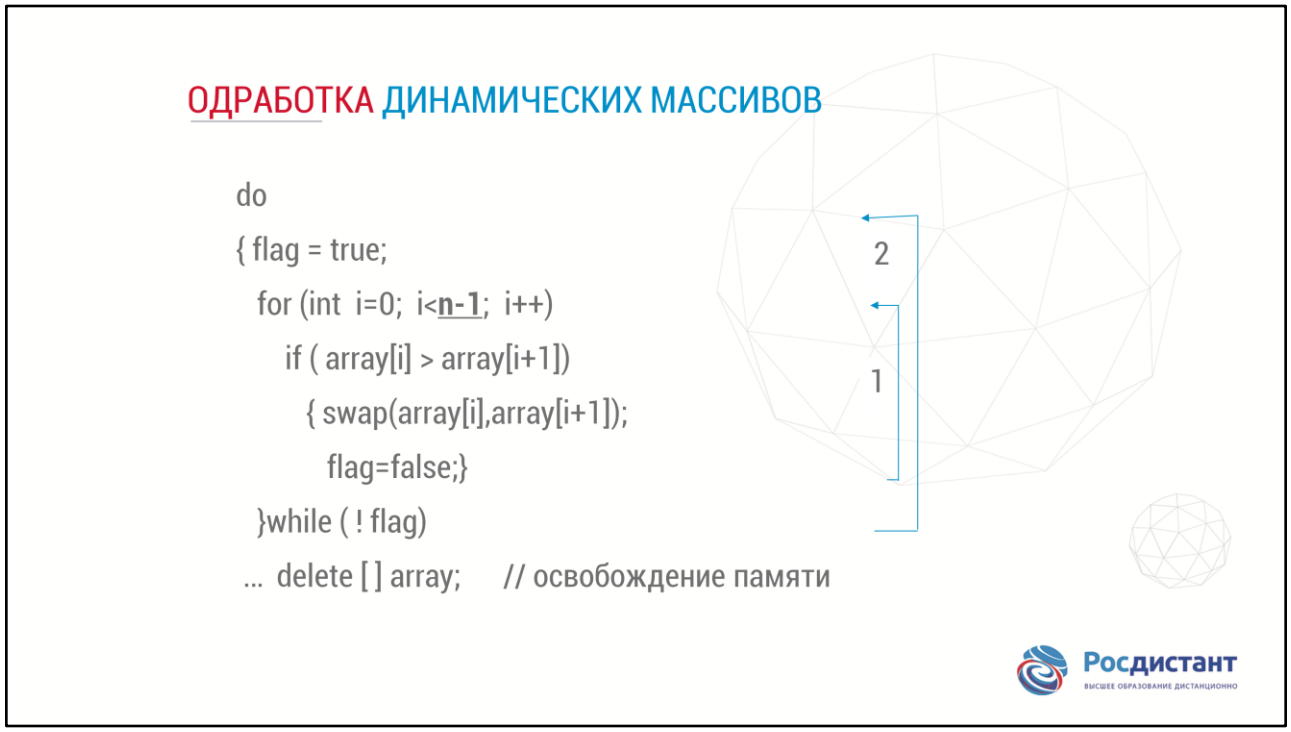
Используем динамический массив. Размер массива заранее не определен. С помощью оператора **new** описан динамический массив**.**

Согласно этому описанию, выделяется память под хранение элементов массива. Заметим, что обнуление памяти при ее выделении не происходит. Инициализировать динамический массив нельзя. Обратится к элементу динамического массива можно как обычно с помощью индекса, так и с использованием указателя. Дело в том, что в переменной указателе хранится

адрес начала массива. Размер выделенной памяти под размещение массива, зависит от типа элементов массива и размера массива, то есть от количества элементов.

Каждый раз, запуская программу на выполнение, вводится размер массива и элементы массива с клавиатуры.

Для реализации процесса сортировки введена дополнительная переменная **flag.** Тип переменной определен как логический. Эта переменная будет использована как признак упорядоченности элементов массива. Напомним, логические переменные могут принимать одно из двух значений: истина или ложь. Исходя из этих свойств логических переменных и построен алгоритм решения поставленной задачи.



Слайд 119

Для сортировки элементов массива используется структуру вложенных циклов.

Введены обозначения:

1 – внутренний цикл;

2 – внешний цикл.

Внутренний цикл, реализованный оператором цикла с параметром, позволяет один раз пройтись по массиву, сравнивая соседние элементы.

Обратите внимание на конечное значение параметра внутреннего цикла в операторе **for.** На слайде это выражение подчеркнуто. Последний раз в цикле сравнивается предпоследний и последний элементы массива.

Перед каждым прохождением по массиву, переменной **flag** присваивается значение истина, предполагая, что массив упорядочен. Если предыдущий элемент, окажется больше последующего, эти элемента меняются местами и выставляется признак того, что массив еще не отсортирован, переменной **flag** присваивается значение ложь.

Внешний цикл, который описан с помощью оператор цикла с постусловием, продолжается до тех пор, пока все элементы не будут упорядочены по возрастанию.

Если необходимо упорядочить элементы по убыванию, достаточно в этом

фрагменте, в структуре оператора проверки условия, поменять знак больше на знак меньше.

Если массив отработан, можно освободить память, отведенную под массив с помощью оператора **delete**.

Для обмена элементов местами будем использовать функция **swap**. У этой функции два аргумента - соседние элементы массива, значения которых меняются местами.

Слайд 120

При решении определенных задач, одной из главных моментов является вопрос, как структурировать информацию, то есть правильная организация информации.

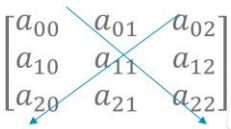
Двумерный массив можно рассматривать как матрицу с определенным количеством строк и столбцов.

Весь набор информации объединяется общим именем. Доступ к отдельным элементам осуществляется с помощью двух индексов, номером строки и номером столбца, на пересечении которых находится этот элемент. Причем, первый индекс, используемый при обращении к элементам массива, всегда номер строки, а второй всегда номер столбца. Индексами двумерного массива могут быть переменные, выражения или константы целого типа.

Двумерные массивы должны быть описаны в программе, указан тип элементов и количество элементов, что определяет объем выделяемой памяти под хранение элементов массива.

Надо помнить, что в двумерном массиве всегда есть нулевая строка и нулевой столбец.

ДВУМЕРНЫЕ МАССИВЫ. МАТРИЦЫ



побочная диагональ главная диагональ

Полезные формулы:

i=j	элементы главной диагонали
i<j	элементы под главной диагональю
i>j	элементы над главной диагональю

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 121

В двумерных массивах есть понятие главной и побочной диагоналей. На слайде обозначены эти диагонали.

При решении многих задач бывает полезно знание некоторых соотношений между индексами массивов.

Двумерный массив можно рассматривать как матрицу. Если количество строк равно количеству столбов, матрица называется квадратной.

В таблице приведены формулы, которые определяют соотношения между индексами элементов, которые расположены на главной диагонали. Соответственно, определены соотношения индексов элементов, расположенных над главной диагональю и под главной диагональю. Знание этих соотношений между индексами двумерного массива, помогает в решении многих задач.

При обозначении индексов элементов двумерного массива на первом месте всегда указывается – номер строки, на втором месте – всегда указывается номер столбца.

ИНИЦИАЛИЗАЦИЯ ЭЛЕМЕНТОВ МАССИВА

Описание и инициализация массива	Присвоенные значения
1. int a[3] [2] ={{ 1 }, { 2 }, { 3 }};	a[0][0] =1 a[0][1]=0 a[1][0] =2 a[1][1]=0 a[2][0] =3 a[2][1]=0
2. float b[ ] [2] = {{1}, {2}};	b[0][0]=1 b[0][1]=0 b[1][0]=2 b[1][1]=0

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 122

На слайде приведены примеры описания и определения элементов двух массивов. Обратите внимание, последовательность значений, которые присваиваются элементам массива, регулируется фигурными скобками и запятыми между фигурными скобками.

Известно, что на языке **C ++,** переменные, описанные в программе, автоматически не обнуляются. Но при этом надо помнить, что достаточно определить значения только некоторых элементов массива. Тогда остальные элементы массива будут автоматически обнулены.

В первом примере описан двумерный массив целых чисел. Определены значения элементов первого столбца. Распределение значений элементов обеспечивается с помощью фигурных скобок.

Во втором примере описан массив вещественных чисел. Обратите внимание, что количество строк в явном виде не указано, о чем свидетельствуют пустые квадратные скобки рядом с именем массива. Это вполне допустимо. Количество строк будет определено присвоенными значениям элементам массива. Количество строк во втором массиве - равно двум.

В нижней строке таблицы приведены значения, присвоенные элементам массива.

ВВОД-ВЫВОД ЭЛЕМЕНТОВ ДВУМЕРНОГО МАССИВА

Ввод элементов массива по строкам	Ввод элементов массива по столбцам

for (i=0; i<n; i++)
for (j=0; j<m; j++)
cin>>a[i][j];
	for (j=0; j<m; j++)
	for (i=0; i<n; i++)
	cin>>a[i][j];

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 123

На слайде приведены два варианта ввода элементов двумерного массива: по строкам и по столбцам.

Не важно, как вводятся элементы массива, по строкам или по столбцам. Необходимо знать, что элементы массива хранятся в памяти по строкам. То есть, сначала элементы первой строки, затем второй строки и так далее.

Ввод элементов двумерного массива выполняется во вложенных циклах. Способ ввода элементов массива по строкам или по столбцам обусловлен законом изменения индексов строк и индексов столбцов во внутреннем и внешнем циклах.

Известно, что во вложенных циклах, индекс внешнего цикла меняется медленнее, чем индекс внешнего. При одном значении параметра внешнего цикла, параметр внутреннего пробегает все свои возможные значения. Именно это правило и позволяет организовать ввод элементов двумерного массива по строкам или по столбцам.

В двумерных массивах всегда присутствует нулевая строка и нулевой столбец. На следующих слайдах приведены примеры обработки двумерных массивов.

Слайд 124

На слайде сформулирована задача.

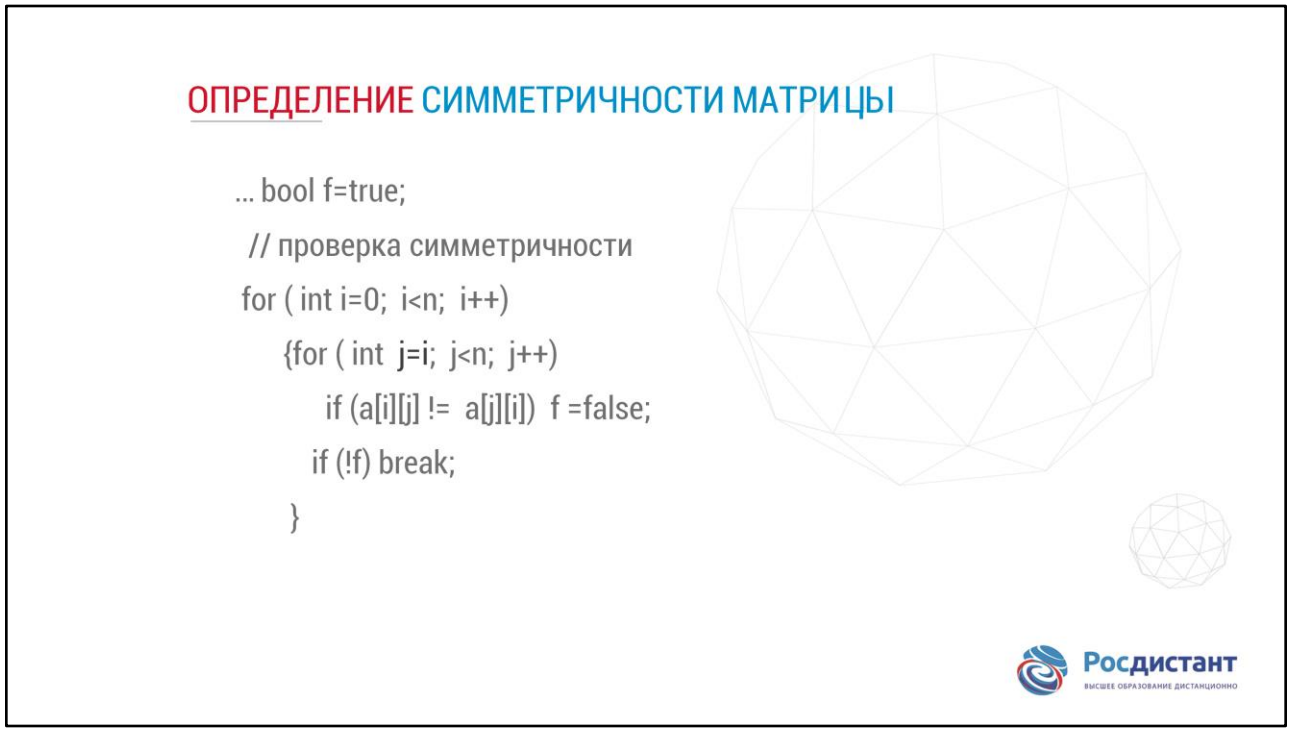
Симметричной считается матрица, у которой элементы по разные стороны главной диагонали равны, то есть если мысленно сложить матрицу по линии главной диагонали, элементы должны совпасть.

На слайде приведена квадратная матрица. И проанализировав индексы элементов по разные стороны главной диагонали, определяются, какие элементы подлежат сравнения. В сравнении участвуют элементы, у которых индексы зеркально меняются.

Пример симметричной матрицы с конкретными числами приведен на слайде.

Введем переменную логического типа, которая будет принимать значение - истина, если матрица симметрична, и принимать значение ложь, если матрица не симметрична.

Перебор элементов матрицы будет производиться во вложенных циклах. Для перебора элементов двумерного массива удобно использовать оператор цикла с параметром, который автоматически будет менять индексы массива.



Слайд 125

На слайде приведен фрагмент программы, позволяющий определить симметричность квадратной матрицы. Обратите внимание, и внутренний цикл и внешний цикл повторяются **n** раз.

Переменная **f** определена как переменная логического типа. Логические переменные могут принимать одно из двух значений, истина или ложь. Первоначально, переменной **f** присвоено значение **true -** истина. Предполагаем, что матрица симметрично.

Перебор элементов массива, расположенных по разные стороны от главной диагонали, производится во вложенных циклах. Если, на каком-то шаге итерации, элементы, расположенные по разные стороны от главной диагонали, не совпали, логической переменной **f** присваивается значение **false** - ложь.

После завершения внутреннего цикла анализируется значение переменной **f.** Если значению этой переменной на коком-либо шаге итерации внутреннего цикла было присвоено значение **false** – ложь, то нет смысла продолжать проверки. Значит матрица не симметрична и внешний цикл прерывается оператором **break.** Вспомним, что оператор **break** прерывает только тот цикл, внутри которого описан, то есть прерывается внешний цикл.

Оптимизируем процесс вычислений. Обратите внимание на закон изменения параметра внутреннего цикла **j**, выделенное выражение на слайде. Этот факт

позволяет сократить количество вычислений, то есть элементы верхнего треугольника сравниваются с элементами нижнего треугольника относительно главной диагонали.

ОПРЕДЕЛЕНИЕ СИММЕТРИЧНОСТИ МАТРИЦЫ

```
for ( i=0;  i<n;  i++)
  {  for ( j=0;  j<n;  j++)
        cout<<a[ i ][ j ]<<"\t";
    cout<<endl; }
  if (f) cout <<"\n Симметрична";
     else  cout <<"\n Не симметрична"; ...
```

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 126

В результате вычислений на экране распечатается сообщение о симметричности или несимметричности матрицы.

На слайде приведен фрагмент, распечатывающий исходный массив в виде матрицы. Печать массива реализована во вложенных циклах.

Внешний цикл осуществляет переход от строки к строке.

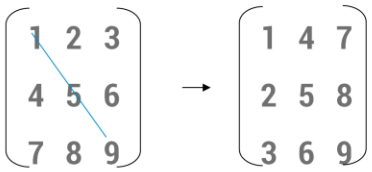
Внутренний цикл выводит элементы одной отдельно взятой строки. Курсор удерживается на данной строке экрана, пока выводятся элемента одной строки матрицы. Между элементами используется табуляция, управляющий символ **\t**, чтобы числа не сливались.

После вывода каждой строки осуществляется перевод курсора на начало новой строки. Для перевода курсора на начало следующей строки использована функция **endl.**

Анализируется значение переменной **f.** И в зависимости от значения этой переменной, выдается сообщение о симметричности матрицы.

ТРАНСПОНИРОВАНИЕ МАТРИЦЫ

Транспонирование матрицы	Фрагмент программы

	...
	for (i=0; i<n; i++)
	for (j=i; j<n; j++)
	swap(a[i][j],a[j][i]);
	...

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 127

На слайде приведен фрагмент, демонстрирующий решение задачи транспонирования матрицы. Фактически, необходимо поменять местами элементы строк с элементами столбцов.

При решении этой задачи можно использовать подходы предыдущего примера, определения симметричности матрицы.

Для перебора все элементов матрицы использованы вложенные циклы. Обратите внимание на подчеркнутый фрагмент в законе изменения параметра внутреннего цикла **j**. Если в предыдущей задаче это действие только сокращает количество проверок, оптимизирует процесс вычислений, то в данном случае – это необходимо. Иначе – матрица останется неизменной.

Фактически элементы верхнего треугольника, образованного главной диагональю, меняются местами с элементами нижнего треугольника квадратной матрицы. Индексы элементов верхнего и нижнего треугольника зеркально меняются.

Обмен выполняется с помощью функции **swap.**

Слайд 128

Рассмотрим задачу:

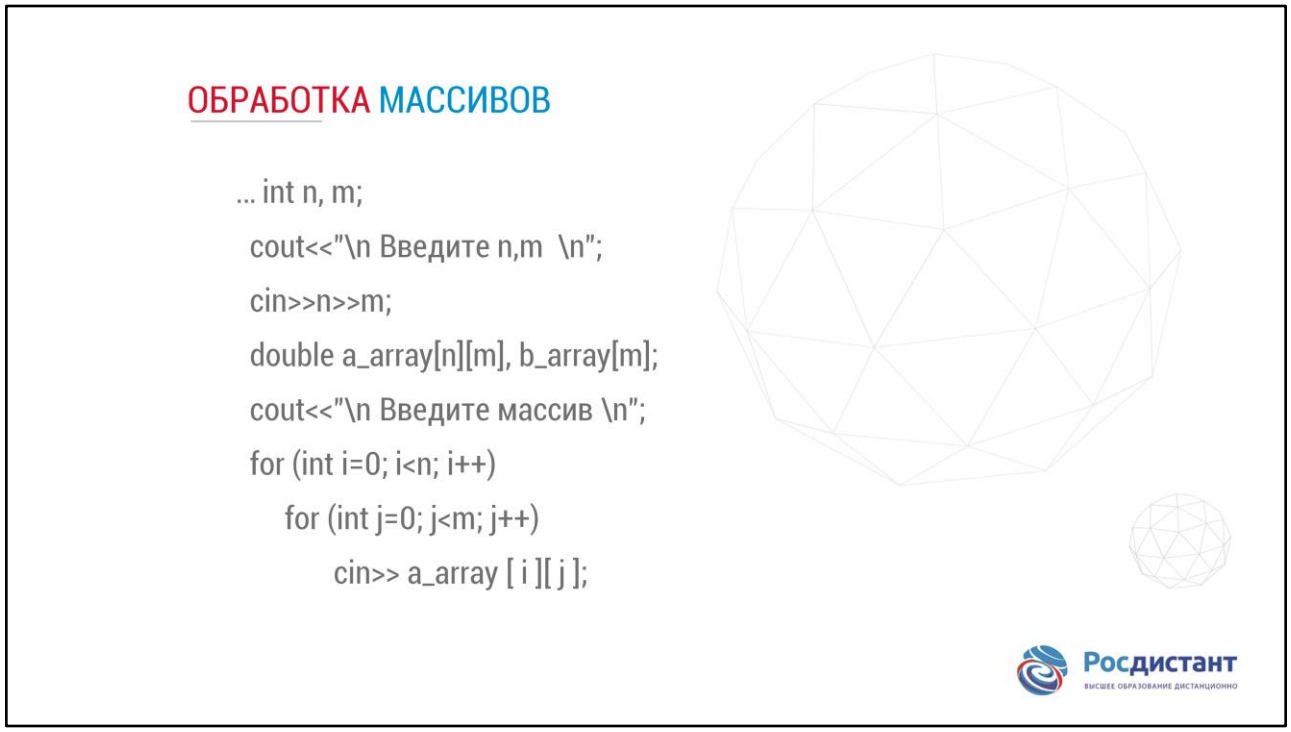
требуется сформировать одномерный массив, каждый элемент которого равен произведению элементов соответствующего столбца двумерного массива, содержащего вещественные числа.

Согласно формулировке задачи, при описании массива необходимо выбрать тип **float** или **double** для элементов массива.

На слайде приведены: конкретный пример исходного двумерного массива с именем **a_array** и пример сформированного из исходного одномерного массива с именем **b_array**.

Исходные данные – двумерный массив. Значения элементов двумерного массива необходимо ввести с клавиатуры.

Для наглядности полученных результатов, распечатать исходный массив в виде матрицы. Под матрицей распечатать элементы одномерного массива, причем каждый элемент вывести на печать под соответствующим столбцом двумерного массива.



Слайд 129

Размер одномерного массива равен количеству столбцов в двумерном массиве.

Известно, что на языке **C++** описанные переменные автоматически не обнуляются. Более того, при подсчете произведения необходимо присвоить первоначально элементам одномерного массива значения, равные единице. В противном случае, результат будет искажен.

Описаны два массива вещественного типа. Тип одномерного массива совпадает с типом элементов двумерного массива. Размер одномерного массива зависит от количества столбцов двумерного массива.

Размер двумерного массива, количество строк и количество столбцов, вводится с клавиатуры. Обратите внимание, массив описан после ввода его размера. Для тестирования и отладки программы можно задавать небольшие размеры массива, после чего можно использовать данную программу для обработки массивов большей размерности.

Ввод элементов двумерного массива осуществляется во вложенных циклах по строкам.

Обратите внимание, переменные, которые являются параметрами цикла, индексами двумерного массива, описаны в структуре оператора цикла **for.**

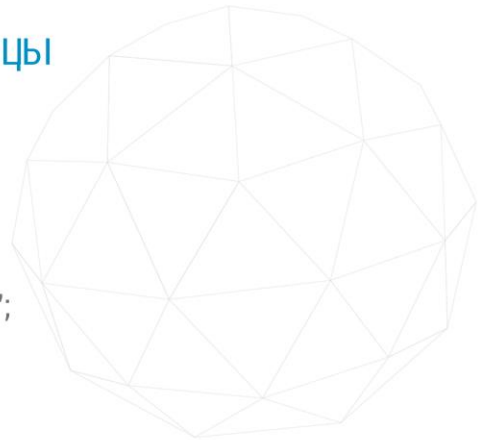
Следовательно эти переменные являются локальными и не доступны вне этого

оператора. При повторном обращении к этим переменным их придется повторно описать. На следующем слайде продемонстрирован этот факт.

Индексы массива меняются с нуля. Напоминаем, что в массиве есть нулевая строка и нулевой столбец.

ПЕЧАТЬ МАССИВА В ВИДЕ МАТРИЦЫ

```
cout<<"\n Исходный массив\n";
for (int j=0; j<m; j++)
    {  for (int i=0; i<n; i++)
            cout<<a_array [i][j]<< "\t";
       cout<<"\n";
     }
```



Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 130

На слайде – фрагмент программы, позволяющий вывести двумерный массив в виде матрицы. Вывод организован во вложенных циклах.

Во внутреннем цикле выводятся элементы одной отдельно взятой строки. Числа одной строки выводятся на одной строке экрана. Для того чтобы числа на слипались, используется управляющий символ **\t –** управляющий символ табуляция**.** Во внутреннем цикле всего один оператор, вывод на печать, поэтому при построении тела вцикла фигурные скобки можно опустить.

Внешний цикл позволяет переходить от строки к строке. Во внешнем цикле два оператора:

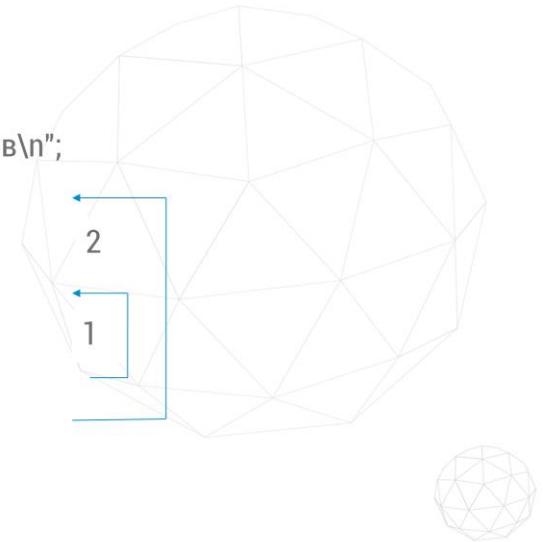
цикл вывода элементов одной строки;

- перевод курсора на начало следующей строки.
- Поэтому тело внешнего цикла заключено в фигурные скобки.

Элементы одномерного массива надо вывести так, чтобы они четко располагались под соответствующим столбцом двумерного массива.

ФОРМИРОВАНИЕ МАССИВОВ

```
cout<<"\n Сформированный массив\n";
  for (int j=0; j<m; j++)
     {  b_array [j]=1;
        for (int i=0; i<n; i++)
             b_array [j]*= a_array [i][j];
        cout<< b_array [j]<<"\t "; }
  …
```



Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 131

На слайде продемонстрирован фрагмент формирования одномерного массива. Используется структура вложенных циклов.

Введены обозначения:

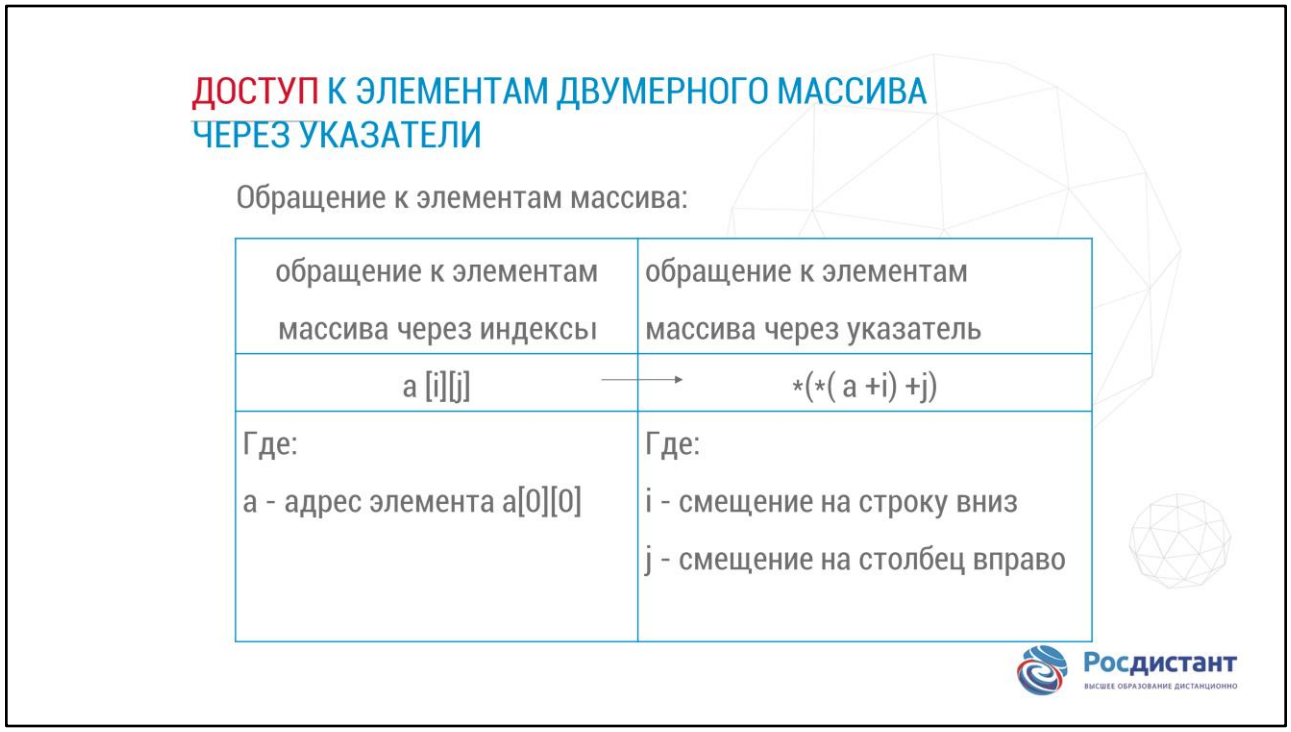
1 – внутренний цикл;

2 – внешний цикл.

Во внутреннем цикле, параметром которого является индексом строки **i ,** подсчитывается произведение элементов одного отдельного столбца. Фактически формируется элемент одномерного массива.

Во внешнем цикле осуществляется переход от столбца к столбцу, и сформированное значение элемент одномерного массива выводится на печать. В начале внешнего цикла, перед формированием значения элемента одномерного массива, ему присваивается значение равное единице. Если этого не сделать, не известно какие значения присвоятся элементам одномерного массива. Переменной, в которой накапливается произведение, необходимо предварительно присвоить значение равное единице.

При использовании вложенных циклов, мысленно соедините начало и конец циклов, и убедитесь, что они не пересекаются.



Слайд 132

Мы уже познакомились с понятием переменных-указателей. Доступ к элементам двумерного массива можно осуществит с помощью операций с указателями. Еще раз вспомним, что имя массива - указатель на начало области, отведенной под хранение элементов массива.

На слайде показаны две идентичные записи обращения к элементам двумерного массива. Обращаться к элементам двумерного массива можно и с помощью двух индексов, как показано в левом столбце. Причем, первый индекс всегда номер строки, а второй индекс всегда номер столбца на пересечении которых расположен данный элемент. Переменные, которые служат индексами массива, должны иметь целочисленный тип.

Идентичная запись обращения к элементам массива приведена в правом столбце. Ранее мы уже рассматривали переменные-указатели. Имя массива – это, фактически, указатель на начало области, выделенной под хранение элементов массива. Наличие круглых скобок при обращении к элементам массива через указатели обусловлено синтаксисом языка **C++ .** Внутренние скобки демонстрируют смещение, переход от строки к строке. Внешние скобки позволяют переходить от элемента к элементу в пределах одной строки. Наличие двух звездочек при обращении к элементам двумерного массива необходимо.

Далее приведены примеры обработки массивов с помощью указателей.

Слайд 133

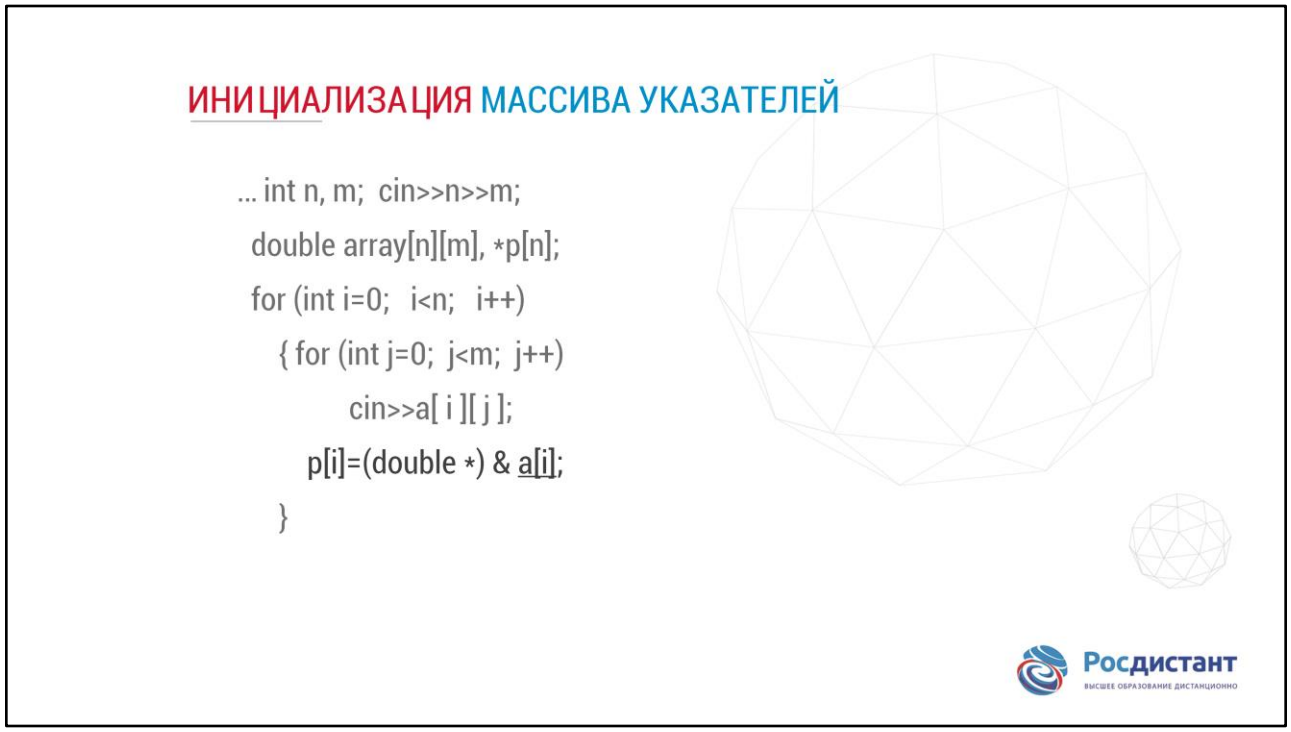
Рассмотрим задачу: составить программу, которая выведет на экран элементы двумерного массива в порядке возрастания первого элемента каждой строки. Главное условие при решении данной задачи - исходный массив должен остаться неизменным.

Допустим, результаты сессии студентов содержатся в двумерном массиве. Каждая строка - это результаты сессии студента, соответствующие его номеру в списке группы. Список группы студентов отсортирован в алфавитном порядке, не нарушая списка вывести на печать результаты сессии в порядке возрастания успеваемости студентов. Вот ситуация, когда необходимо решить задачу, озвученную выше. Исходный массив с результатами сессии должен остаться неизменным.

Для решения этой задачи будем использовать массив указателей. Размер массива указателей совпадает с количеством строк двумерного массива. Элементами массива указателей будут адреса первых элементов каждой строки. Сравнивать будем первые элементы каждой строки исходного массива, а менять местами значения элементов массива указателей, то есть адреса первых элементов. Фактически сортируются элементы массива указателей. Сортировка будет зависеть от элементов исходного массива.

Печать двумерного массива осуществляется через массив указателей.

На слайде приведены примеры исходного массива и, как должен выглядеть на экране, результат. Также показан массив указателей.



Слайд 134

На слайде представлен фрагмент, в котором определены исходный двумерный массив и одномерный массив указателей. Размер одномерного массива указателей равен количеству строк исходного двумерного массива.

Напоминаем, указатели должны иметь тип информации, на которую они указывают. Следовательно, если исходный массив имеет тип **double,** то и тип массива указателей описан через тип **double**. Символ ***** при описании определяет, что это массив указателей.

Во внутреннем цикле вводятся элементы массива по строкам. После выхода из внутреннего цикла адрес первого элемента каждой строки присваивается соответствующему элементу массива указателей. Адрес определяется с помощью унарной операции **&**.

Обратите внимание на подчеркнутый и выделенный фрагмент – это и есть первый элемент в каждой строке, адрес которого присваивается соответствующему элементу массива указателей. Имя массива указатель. Задавая только один индекс при обращении к двумерному массиву, мы фактически определяем адрес, по которому в памяти располагается первый элемент соответствующей строки. Выше мы подробно рассматривали, что такое имя массива. Поэтому такая запись вполне допустима!

СОРТИРОВКА ЧЕРЕЗ МАССИВ УКАЗАТЕЛЕЙ

```
... bool flag;
do { f lag=true;
      for (i=0; i<n-1;  i++)
         if (*p[i]>*p[i+1] )   // сравнение чисел
           {swap(p[i],p[i+1]);  // обмен адресов
            flag=false; }
 } while( ! flag );
```

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 135

Введем дополнительную переменную **flag**. Она используется как признак при сортировке массива адресов, но не сами адреса анализируются, а анализируются элементы массива, расположенные по этим адресам. Переменная **flag** имеет логический тип. Эта переменная может принимать одно из двух значений: истина или ложь.

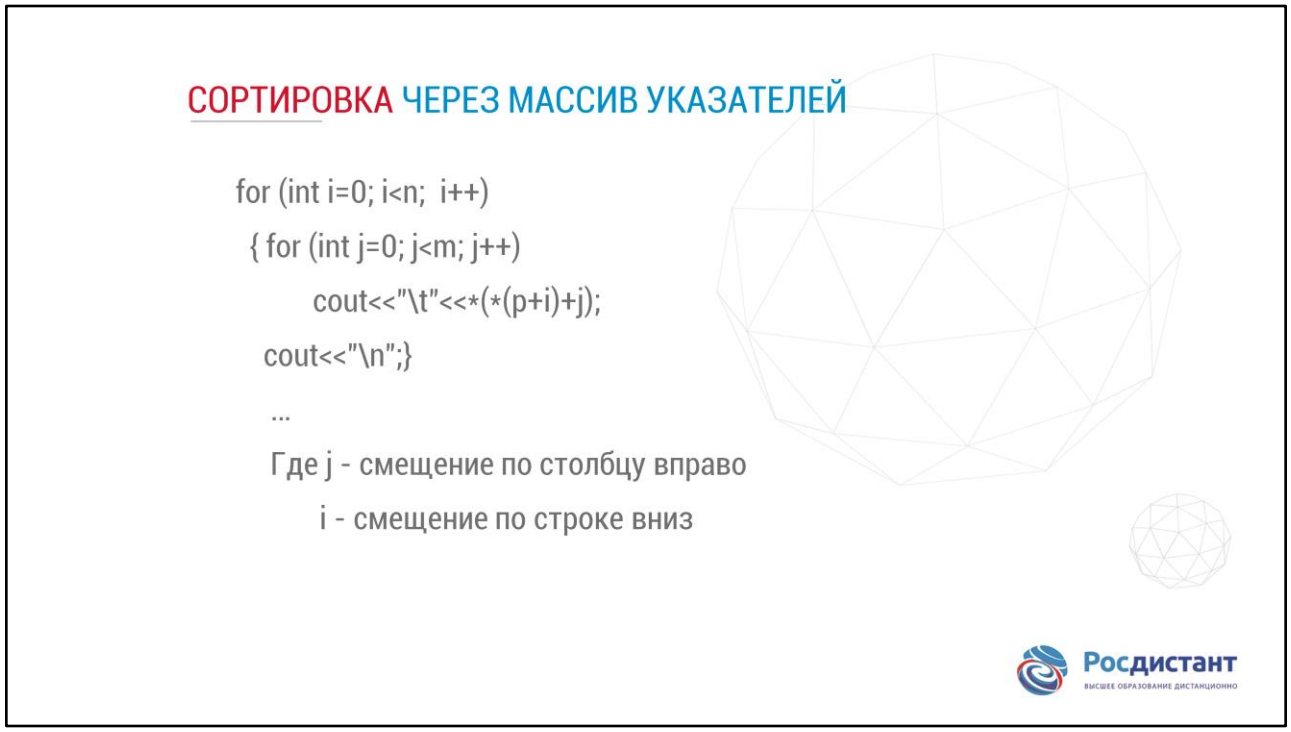
Чтобы получить доступ к числам, при сравнении используется символ ***,** то есть сравниваются первые элементы строк исходного массива, а меняются местами адреса в массиве указателей.

Для обмена используется функция **swap**, которая была рассмотрена выше.

Сортировка реализуется во вложенных циклах. Внешний цикл, цикл с постусловием, продолжается до тех пор, пока значение переменной **flag** - ложно.

Условием прекращения внешнего цикла - является значение переменной **flag,** равное истине, то есть имеет значение **true,**

После того, как массив указателей отсортирован, цикл прекращается.



Слайд 136

Следующий фрагмент демонстрирует вывод двумерного массива через массив указателей.

При этом исходный массив остался неизменным.

Используя индексы **i** и **j**, осуществляется переход от одного элемента к другому через указатели. Где индексы:

j – определяет переход от столбца к столбцу, то есть переход от элемента к элементу одной строки;

i – определяет переход от строки к строке.

Напомним, элементы массива располагаются в памяти по строкам, не зависимо от того, как был организован ввод элементов исходного массива.

Переменная-указатель **p** описана как указатель на информацию вещественного типа, так как элементы массива имеют вещественный тип. Поэтому увеличение значения переменной-указателя на единицу соответствует переходу к следующему элементу массива, соответствует перемещение указателя на восемь байт, так как под тип **double** выделяется восемь байт памяти. Массив выведется на экран в виде матрицы.

На этом мы заканчиваем тему массивы. Однако, с массивами нам придется столкнуться и при изучении следующих разделов нашего курса: при

использовании функций, в теме структуры, в разделе символьные массивы и так далее.

Введение в языки программирования C и C++ | Уроки C++ - Ravesli / https://ravesli.com/urok-2-vvedenie-v-yazyki-programmirovaniya-c-i-s/

Т, А. Павловская С/С++ Программирование на языке высокого уровня/http://cph.phys.spbu.ru/documents/First/books/7.pdf

Введение в программирование | Уроки C++ - Ravesli/https://ravesli.com/urok-1-vvedenie-v-programmirovanie/

Технология программирования - Информатика, автоматизированные информационные технологии и системы/ https://studref.com/441961/informatika/tehnologiya_programmirovaniya

Бьерн Страуструп. Язык программирования С++ 11/ https://vk.com/doc-145125017_450056359

Язык СИ++ Учебное собие / http://5fan.ru/wievjob.php?id=4301

А.А. Шелупанов, В.Н. Кирнос ИНФОРМАТИКА. БАЗОВЫЙ КУРС Учебник в четырех частях. /https://edu.tusur.ru/publications/521/download