В этом подразделе я попытаюсь дать ответ на вопрос, что такое массив. В шестом уроке учебника, мы с вами рассматривали [понятие переменной](http://iguania.ru/tutorial/chapter-6), как самостоятельного контейнера, хранящего значение определенного типа (int - целые, float - дробные с точностью 6-7 знаков после запятой, double - дробные с точностью 13-14 знаков, char - символьные и так далее). Каждый тип переменной хранит данное своего типа и занимает определенный объем оперативной памяти компьютера. К примеру, int занимает в памяти 4 байта. Для того, чтобы лучше понять, каким образом переменные располагаются в памяти, просмотрите мою статью "[Устройство оперативной памяти](http://iguania.ru/tutorial/chapter-13)". В программировании часто приходится сталкиваться с однотипными объектами-переменными. Любое высказывание лучше понять на примере. Например, вот такая программа:

//Программа запрашивает оценки по экзамену,

//сохраняет их и затем выводит на печать

#include <iostream>

using namespace std;

int main()

{

int point\_1, point\_2, point\_3, point\_4, point\_5;

cout << "Введите оценку 1-го учащегося за экзамен по информатике: ";

cin >> point\_1;

cout << "Введите оценку 2-го учащегося за экзамен по информатике: ";

cin >> point\_2;

cout << "Введите оценку 3-го учащегося за экзамен по информатике: ";

cin >> point\_3;

cout << "Введите оценку 4-го учащегося за экзамен по информатике: "

cin >> point4;

cout << "Введите оценку 5-го учащегося за экзамен по информатике: ";

cin >> point5;

cout << endl << "Были введены следующие оценки учащихся" << endl;

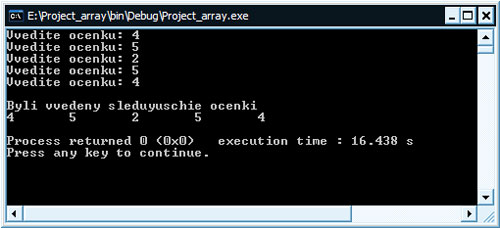
cout << point\_1 << "\t" << point\_2 << "\t" << point\_3

<< "\t" << point\_4 << "\t" << point\_5 << endl;

return 0;

}

Результат работы программы:



Программа, приведенная выше, запрашивает у пользователя пять оценок, которые сохраняются в пяти переменных, которые мы предварительно объявили

int point1, point2, point3, point4, point5;

В конце мы выводим эти оценки на экран, табулируя результаты с помощью оператора "\t".

А теперь, я ниже приведу вариант этой же программы, но с использованием массива. Вот, что получится:

//Программа запрашивает оценки по экзамену,

//сохраняет их и затем выводит на печать

#include <iostream>

using namespace std;

int main()

{

int point[5];

for(int i = 0; i < 5; i++)

{

cout << "Введите оценки пяти учащихся за экзамен по информатике: ";

cin >> point[i];

}

cout << endl << "Были введены следующие оценки учащихся за экзамен по информатике" << endl;

for(int i = 0; i < 5; i++)

cout << point[i] << "\t";

return 0;

}

Эта программа выведет абсолютно такой же результат, но вот построена она уже иначе. Такой вариант реализации более удобный и компактный за счет того, что мы используем массив. Вот его объявление

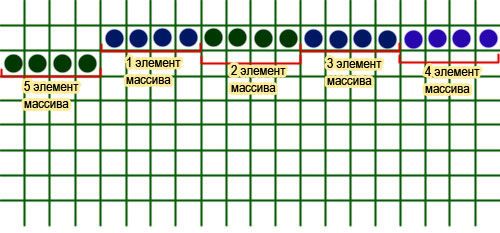
int point[5];

И оно логически равноценно этому

int point1, point2, point3, point4, point5;

Почему? Потому что массив *point* содержит в себе пять элементов типа *int*.

Вот как это будет выглядеть в памяти компьютера:



Как видите, переменные, размером в четыре байта (тип int) идут в оперативной памяти компьютера подряд. Совокупность этих переменных одного типа и есть массив. И переменные эти именуются как элементы массива.

Важно!

Нумерация элементов массива начинается с нуля!!! То есть, если мы имеем массив, состоящий из пяти элементов, то доступ к ним можно получить таким образом:

point[0];

point[1];

point[2];

point[3];

point[4];

Рассмотрев, применение массива на небольшом практическом примере, я все же хочу дать определение и ответить на вопрос: что такое массив?

*Массив - это* совокупность переменных, содержащих данные одного типа (например, int), объединенных одним общим именем. Каждая отдельно взятая переменная, называется элементом (ячейкой) массива. В памяти элементы массива всегда располагаются строго последовательно, благодаря чему повышается скорость доступа к данным. Доступ к элементам массива осуществляется через указание индексов, которые указываются после имени в квадратных скобках и обязательно нумеруются, начиная с нуля.

# Одномерные массивы

В этой главе учебника поговорим об одномерных массивах, как о самых простых. Что такое одномерный массив? Это массив, у которого есть лишь один параметр - это количество элементов. Эти элементы расположены последовательно, т.е. идут друг за другом.

Еще раз повторим то, как объявляются и инициализируются массивы, в нашем случае одномерные массивы.

Допустим, что нам нужно объявить одномерный массив, состоящий из 8-ми элементов и присвоить его элементам значения, вычисленные по формуле: номер элемента массива, умноженный на 2 и минус 1. Т.е. для 3-го элемента будет 3 \* 2 - 1 = 5. Смотрим

//Одномерный массив

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

//объявляем массив из 8-ми элементов

int array[8];

//инициализируем массив данными по формуле

for(int i = 0; i < 8; i++)

array[i] = i \* 2 - 1;

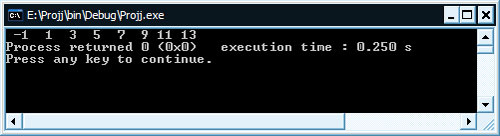
//выводим массив на экран

for(int i = 0; i < 8; i++)

cout << setw(3) << array[i];

return 0;

}



Как видите, ничего сложного в работе с одномерными массивами нет.

//объявляем массив из 8-ми элементов

int array[8];

Объявляется массив путем указания типа его элементов, в нашем случае это целые (int), затем указывается его имя, в нашем случае это array, а затем в квадратных скобках указывается количество элементов, в нашем случае их восемь. Итак, память для массива мы зарезервировали и на этапе компиляции программы она будет выделена (8 \* 4 байта = 32 байта, т.к один элемент типа int занимает в памяти 4 байта). Но в массиве на данный момент содержится различный "мусор", т.е. различные значения, которые возможно применялись в других программах и так далее. Нужно изменить (задать) эти значения, т.е. выполнить инициализацию массива. Проще всего ее выполнять с помощью цикла, в котором мы последовательно проходим по всем элементам массива. Инициализируем, применяя формулу i \* 2 - 1, где i - значение индекса массива.

//инициализируем массив данными по формуле

//инициализируем массив данными по формуле

for(int i = 0; i < 8; i++)

array[i] = i \* 2 - 1;

Теперь каждый элемент нашего одномерного массива содержит свое заданное значение. Хотелось бы их просмотреть, а для этого нам нужно подумать, как вывести их на экран. Для этого опять же проще всего использовать цикл, в котором мы будет последовательно проходить по всем элементам массива, выводя их на экран в одну строку. Для каждого значения мы выделим на экране место в 3 ячейки, для этого будем использовать манипулятор вывода потока setw с параметром 3.

//выводим массив на экран

for(int i = 0; i < 8; i++)

cout << setw(3) << array[i];

### Одномерные массивы - решение задач

Теория - это хорошо, но практика лучше. Немного теории об одномерных массивах я вам рассказал, а теперь давайте займемся практикой решения задач.

#### Одномерные массивы - задача № 2

Создать одномерный массив, содержащий 15 элементов, наполнить его случайными значениями в интервале от 1 до 200. Посчитать и вывести:

1. Сумму всех четных элементов массива.
2. Сумму всех нечетных элементов массива.
3. Сумму всех элементов массива.
4. Произведение всех элементов массива, значения которых меньше 50.
5. Произведение элементов массива с индексами от 2 и 7.

Давайте во всех тонкостях программы разбираться по порядку, даже, несмотря на то, что некоторые моменты нам знакомы, заодно и повторим выученное.

Итак, для начала объявим массив и пять переменных, которые будут нам служить для хранения результатов работы с элементами массива

const int size = 15;

int massiv[size];

int sumChet = 0, sumNechet = 0, sum = 0, proizv1 = 1, proizv2 = 1;

Как мы увидим позже, размер массива часто используется в различных моментах кода программы (например, в циклах прохода по массиву), поэтому удобнее его объявлять не конкретным числом в самом объявлении массива, а отдельно, как константу. Почему как константу? Потому что размер массива не может изменяться во время работы программы, на то он и статический, а не динамический (работа с динамическими массивами будет рассматриваться позже, это отдельная тема). Для того, чтобы компилятор знал о том, что это константа и не выдавал ошибку во время компиляции программы, нужно обязательно объявлять константное значение, в частности размер массива, со спецификатором const.

Массив мы объявили, теперь нужно выполнить его инициализацию. По заданию его нужно заполнить случайными значениями от 1 до 200. Естественно, для этого нам нужно использовать функцию рандомизации (rand() - вырабатывающую случайным значения по определенному алгоритму). Для того, чтобы функция работала правильно ее нужно использовать в паре с другой функцией (srand() ), позволяющей задавать начальную точку отчета для работы функции рандомизации rand(). Эти две функции описаны в заголовочном файле stdlib.h. Для начальной точки отчета функция srand() использует системное время, поэтому подключаем заголовочный файл time.h.

srand(time(NULL));

Задаем нашему одномерному массиву случайные значения

for(int i = 0; i < size; i++)

massiv[i] = 1 + rand() % 200;

Естественно, это удобнее всего делать в цикле, проходя по всем элементам массива последовательно и присваивая каждому свое случайное значение. Для того, чтобы функция rand() генерировала случайные величины в интервале от 1 до 200, нужно использовать вот такой синтаксис: начальная точка плюс rand(), остаток от деления (знак %) конечная точка.

Теперь начинается самая важная работа - это подсчет нужных сумм и произведений. Опять же для прохода по массиву используем цикл for

for(int i = 0; i < size; i++)

{

if(massiv[i] % 2 == 0)

sumChet += massiv[i];

if(massiv[i] % 2 != 0)

sumNechet += massiv[i];

sum += massiv[i];

if(massiv[i] < 50)

proizv1 \*= massiv[i];

if(i == 2 || i == 7)

proizv2 \*= massiv[i];

}

Как обычно, для прохода по всему одномерному массиву мы используем цикл for в интервале от 0 до size, где size является размером массива. Условие используется строгое, без включения самого size, т.к. нумерация элементов массива начинается с нуля. Т.е., если размер массива равен 15, то индексы его будут от 0 до 14 включительно, что есть 15 штук.

Далее нам по условию нужно найти сумму всех четных элементов массива. Для этого мы будем использовать соответствующее условие выборки

if(massiv[i] % 2 == 0)

sumChet += massiv[i];

Думаю здесь все понятно: если остаток от деления значения элемента на 2 равен нулю, то значит значение этого элемента массива четное и, соответственно, его можно приплюсовывать в общую сумму. Т.к. переменную sumChet мы используем для накопления общей суммы, то при объявлении мы сразу ее инициализировали значением 0. Если вы забыли, то напомню, что это сокращенное выражение sumChet += massiv[i]; в полной форме будет таким sumChet = sumChet + massiv[i];

Далее по условию нам нужно найти сумму всех нечетных элементов массива

if(massiv[i] % 2 != 0)

sumNechet += massiv[i];

Для нахождения этой сумму используется обратное условие, т.е. если остаток от деления не равен 0, то выполняем приплюсовывание суммы в общую переменную накопления.

Сумма всех элементов одномерного массива считается довольно просто

sum += massiv[i];

Нет никаких условий, все элементы массива просто последовательно, поочередно складываются.

if(massiv[i] < 50)

proizv1 \*= massiv[i];

Для подсчета произведения всех элементов, значения которых менее 50 используем соответствующее условие выборки.

И последнее, что нам осталось сделать, это найти произведение элементов массива с индексами 2 и 7. Задаем условие, что если в данный момент мы находимся на элементах массива 2 или (||) 7, то выполнить умножение

if(i == 2 || i == 7)

proizv2 \*= massiv[i];

Ниже привожу полный код данной программы:

//Работа с одномерным массивом

#include <iostream>

#include <stdlib.h>

#include <time.h>

//прототип функции для печати массива

void printArray(int[], const int);

using namespace std;

int main()

{

//объявляем необходимые переменные

const int size = 15;

int massiv[size];

int sumChet = 0, sumNechet = 0, sum = 0, proizv1 = 1, proizv2 = 1;

//задаем начало отчета для рандомизатора rand()

srand(time(NULL));

//наполняем массив случайными величинами

for(int i = 0; i < size; i++)

massiv[i] = 1 + rand() % 200;

//выводим содержимое массива на экран

printArray(massiv, size);

//выполняем необходимые подсчеты в цикле

for(int i = 0; i < size; i++)

{

if(massiv[i] % 2 == 0)

sumChet += massiv[i];

if(massiv[i] % 2 != 0)

sumNechet += massiv[i];

sum += massiv[i];

if(massiv[i] < 50)

proizv1 \*= massiv[i];

if(i == 2 || i == 7)

proizv2 \*= massiv[i];

}

//выводим результаты работы программы

cout << "\nSumma chetnih: " << sumChet << endl;

cout << "Summa nechetnih: " << sumNechet << endl;

cout << "Summa vseh: " << sum << endl;

cout << "Proizvedeniye elementov do 50: " << proizv1 << endl;

cout << "Proizvedeniye elementov ot 0 do 4: " << proizv2 << endl;

return 0;

}

//функция вывода массива на экран

void printArray(int a[], const int s)

{

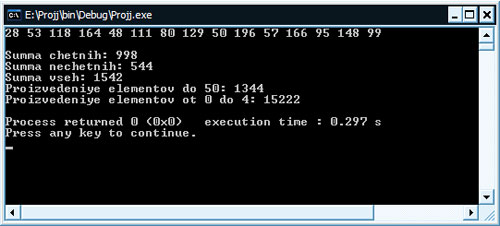
for(int k = 0; k < s; k++)

cout << a[k] << " ";

cout << endl;

}

Результат работы программы будет таким



Далее займемся двумя задачами, которые уже будут несколько сложнее начинающему.

#### Одномерные массивы - задача № 3

Используйте одномерный массив для решения следующей задачи. Компания платит своим продавцам на комиссионной основе. Продавцы получают 200 долларов в неделю плюс 9 процентов от валовой продажи за эту неделю. Например, продавец, валовая продажа которого за неделю составила 5000 долларов, получает 200 долларов плюс 9 процентов от 5000 долларов, или всего 650 долларов. Напишите программу (используя массив счетчиков), которая определяет, сколько продавцов получили заработную плату в каждом из следующих диапазонов (примем допущение, что зарплата каждого продавца округляется до целого значения):

1. $200 - $299
2. $300 - $399
3. $400 - $499
4. $500 - $599
5. $600 - $699
6. $700 - $799
7. $800 - $899
8. $900 - $999
9. $1000 и более.

Попробуйте решить эту задачу самостоятельно, а затем сравнить с моим решением, которое для удобства будет скрыто

Код программы:

//Программа определяет количество продавцов,

//получивших заработную плату из определенного интервала

#include <iostream>

//определяем прототипы функций

void intervals();

void printArray(int[], int);

using namespace std;

int main()

{

//определяем константное число, определяющее размер массива

const int size = 9;

//объявляем массив, инициализируем его нулями

//totalSale - будет запоминать валовую продажу служащего

//general - будет запоминать заработную плату служащего

int array[size] = {0}, totalSale, general;

//выводим интервалы заработков служащих для наглядности

intervals();

//запрашиваем валовую продажу и сохраняем значение

cout << "Total sale (exit -1): " << endl;

cin >> totalSale;

//в цикле запрашиваем количества валовых продаж служащих

while(totalSale != -1)

{

//подсчитываем заработную плату служащего

general = 200 + (totalSale \* 9 / 100);

//организуем подсчет, используя счетчики

if(general >= 200 && general <= 299)

++array[0];

else if(general >= 300 && general <= 399)

++array[1];

else if(general >= 400 && general <= 499)

++array[2];

else if(general >= 500 && general <= 599)

++array[3];

else if(general >= 600 && general <= 699)

++array[4];

else if(general >= 700 && general <= 799)

++array[5];

else if(general >= 800 && general <= 899)

++array[6];

else if(general >= 900 && general <= 999)

++array[7];

else if(general >= 1000)

++array[8];

//запрашиваем валовую продажу и сохраняем значение

cout << "Total sale (exit -1): " << endl;

cin >> totalSale;

}

printArray(array, size);

return 0;

}

//описываем функцию, выводящую на экран интервалы заработков

void intervals()

{

cout << "1. $200 - $299" << endl

<< "2. $300 - $399" << endl

<< "3. $400 - $499" << endl

<< "4. $500 - $599" << endl

<< "5. $600 - $699" << endl

<< "6. $700 - $799" << endl

<< "7. $800 - $899" << endl

<< "8. $900 - $999" << endl

<< "9. $1000 and more" << endl << endl;

}

//описываем функцию, служащую для вывода значений массива

void printArray(int a[], int size)

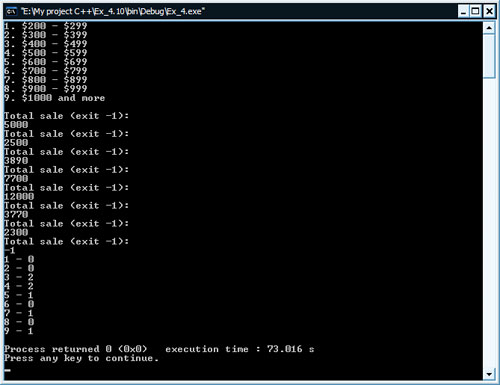
{

for(int i = 0; i < size; i++)

cout << (i + 1) << " - " << a[i] << endl;

}

Результат работы программы



Решим еще одну задачу на одномерные массивы

#### Одномерные массивы - задача №4

Используйте одномерный массив для решения следующей задачи. Прочитайте 20 чисел, каждое из которых находится в диапазоне от 10 до 100 включительно. После того, как прочли очередное число, напечатайте его, но только в том случае, если оно не дублирует ранее прочитанные числа. Предусмотрите "наихудший" случай, когда все 20 чисел различны. Используйте наименьший возможный массив для решения этой задачи.

Код программы:

//Работа с одномерным массивом

#include <iostream>

using namespace std;

int main()

{

//объявляем необходимые переменные

const int size = 20;

int array[size], var;

bool label = true;

//выводим подсказку пользователю

cout << "Enter 20 numbers in an interval from 10 up to 100" << endl << endl;

//начинаем ввод 20-и чисел в цикле

for(int i = 0; i < size; i++)

{

//запрашиваем число, запоминаем его в переменную var

cout << "Enter: ";

cin >> var;

//если число находится в нужном интервале

if(var >= 10 && var <= 100)

{

//записываем его в массив

array[i] = var;

//проверяем, не дублирует ли оно

//введенное ранее число

for(int j = 0; j < i; j++)

{

//если найдено соответствие, то меняем флаг переменной label

if(array[j] == array[i])

label = false;

}

//если соответствия не было найдено, то печатаем число

if(label == true)

cout << array[i] << endl;

}

//в случае, если было введено число не из интервала 10 - 100

else

{

cout << "Error! Incorrect interval" << endl;

i--;

}

//меняем флаг перед входом в новую итерацию

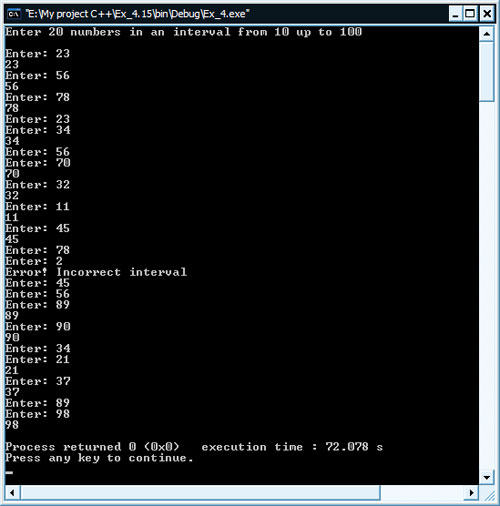
label = true;

}

return 0;

}

Результат работы программы



оворим о двумерных массивах

Массивы в С++ могут быть не только одномерными, т.е. когда данные визуально выстроены в одну линию. Массивы также могут быть и двумерными, трехмерными и так далее. С++ компиляторы поддерживают как минимум 12-ти мерные массивы!!! Естественно, что такими большими массивами на практике никто не пользуется, т.к. человеку сложно их визуально представить у себя в голове, не то что написать программу, которая оперирует таким сложными массивами. На практике редко применяют массивы, более трехмерного. Одномерный - это строка, двумерный - матрица (таблица), трехмерный - куб, а вот дальше уже сложно, поэтому дальше, обычно, никто и не идет... Вот такие дела.

**Двумерный массив** - это так называемая матрица (или таблица), у которой есть строки и столбцы. По соглашению программистов первый индекс массива будет указывать на строки, а второй на столбцы. Вот пример объявления и инициализации двумерного массива, состоящего из трех строк и пяти столбцов.

{{

int aMatrix[3][5] = {{3, 5, 5, 7, 8},

{4, 1, 1, 2, 9},

{3, 8, 8, 9, 7}};

}}

Как видите, двумерный массив имеет два индекса (на то он, в принципе, так и называется). Сразу при объявлении мы его инициализируем целочисленными величинами. Для удобства мы их записываем в виде матрицы (таблицы): каждая строка с новой строки (их у нас 3), в каждой строке 5 столбцов.

Можно и так записать, как показано ниже. Разницы для компилятора не будет никакой. Разве лишь разница будет в визуальном восприятии для человека.

{{

int aMatrix[3][5] = {{3, 5, 5, 7, 8}, {4, 1, 1, 2, 9}, {3, 8, 8, 9, 7}};

}}

Либо вообще так, без указания фигурных скобок, которые логически разделяют строки друг от друга.

int aMatrix[3][5] = {3, 5, 5, 7, 8, 4, 1, 1, 2, 9, 3, 8, 8, 9, 7};

Кстати, последняя запись демонстрирует то, как на самом деле элементы массива размещаются в памяти компьютера. Как я уже писал в предыдущих главах, они идут последовательно.

Если инициализация двумерного массива происходит одновременно с объявлением, то можно даже не указывать первый индекс, т.е. количество строк массива.

{{

int aMatrix[][5] = {{3, 5, 5, 7, 8},

{4, 1, 1, 2, 9},

{3, 8, 8, 9, 7}};

}}

Зная количество столбцов, компилятор при компиляции сам рассчитает количество строк двумерного массива.

Для доступа к элементам двумерного массива нужно, так же, как и для одномерного указать индекс. В данном случае нам нужно будет позаботиться об указании двух индексов. Например, чтобы перезаписать последний элемент второй строки, мы должны использовать такую запись

aMatrix[1][4] = 0;

В этом случае мы перезапишем значение 9 на 0.

Для прохода по двумерному массиву удобнее всего использовать два цикла *for*, вложенных друг в друга. Ниже на примере мы выводим содержимое двумерного массива на экран.

{{

//Работа с двумерным массиво

#include <iostream>

using namespace std;

int main()

{

int aMatrix[3][5] = {{3, 5, 5, 7, 8},

{4, 1, 1, 2, 9},

{3, 8, 8, 9, 7}};

aMatrix[1][4] = 0;

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 5; j++)

{

cout << aMatrix[i][j] << ", ";

}

cout << endl;

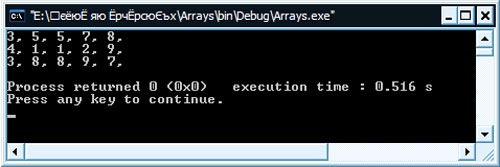
}

return 0;

}

}}

Результат работы программы:



Решение задач на двумерные массивы

Для закрепления пройденного материала мы выполним решение задач на двумерные массивы. Первая задача будет такая:

1. Нужно создать двумерный массив, размером 5 х 7 (пять строк, семь столбцов). Заполнить его случайно целыми числами, в районе от 0 до 30. Вывести получившийся массив на экран. Затем нужно отсортировать строки массива так, чтобы первой шла строка, сумма элементов которой была меньше, чем остальных. И так далее, по возрастанию. Для сортировки удобно использовать, алгоритм [сортировки отбором](http://iguania.ru/algoritmi-programmirovaniya/sortirovka-otborom.html). Вот только в этом случае этот алгоритм сортировки уже будет сортировать не отдельные числа одномерного массива, а строки двумерного массива, исходя из суммы ее элементов.

Попытайтесь сами реализовать программу, выполняющую эту задачу. В случае необходимости можете посмотреть мой вариант решения этой задачи, который привожу ниже.

//Двумерные массивы - решение задач

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

void initMatrix(int[][7], const int, const int);

void replaceColumn(int[][7], const int, int, int);

void printMatrix(int[][7], const int, const int);

int main()

{

const int column = 5, row = 7;

int matrix[column][row];

int counter, result, columnMin;

srand(time(NULL));

//инициализируем двумерный массив случайными величинами,

//затем печатаем его

initMatrix(matrix, column, row);

cout << "\nNoSorted array" << endl << endl;

printMatrix(matrix, column, row);

//делаем проходы столько раз, сколько строк в массиве

for(int N = 0; N < column - 1; N++)

{

result = row \* 30;

//анализируем массив и находим строку,

//сумма элементов которой минимальная

for(int i = N; i < column; i++)

{

//обнуляем счетчик суммы элементов строки

counter = 0;

//проходим по элементам данной строки

//и считаем сумму

for(int j = 0; j < row; j++)

counter += matrix[i][j];

//result в итоге будет хранить значение минимальной суммы

//columnMin будет хранить значение строки, в которой

//была найдена минимальная сумма

if(result > counter)

{

result = counter;

columnMin = i;

}

}

replaceColumn(matrix, row, N, columnMin);

}

//печатаем отсортированный массив

cout << "\nSorted array" << endl << endl;

printMatrix(matrix, column, row);

return 0;

}

//инициализация двумерного массива

void initMatrix(int matrix[][7], const int column, const int row)

{

for(int i = 0; i < column; i++)

for(int j = 0; j < row; j++)

matrix[i][j] = rand() % 30;

}

//меняем строки массива местами

void replaceColumn(int matrix[][7], const int row, int N, int columnMin)

{

int buffer[row];

for(int j = 0; j < row; j++)

{

buffer[j] = matrix[columnMin][j];

matrix[columnMin][j] = matrix[N][j];

matrix[N][j] = buffer[j];

}

}

//печать двумерного массива

void printMatrix(int matrix[][7], const int column, const int row)

{

for(int i = 0; i < column; i++)

{

for(int j = 0; j < row; j++)

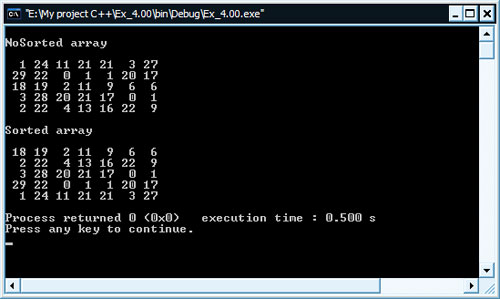
cout << setw(3) << matrix[i][j];

cout << endl;

}

}

Результат работы программы:



2. В качестве второй задачи, предлагаю повторить задумку первой задачи, только на этот раз сортировать не строки по возрастанию, а столбцы двумерного массива. Для начала этого будет достаточно, остальные задачи на двумерные массивы вынесу в практику.

Для начала решим несколько несложных задач на массивы, так сказать для разминки.

#### Задача 1

Нужно написать программу, которая создает одномерный массив, размером в 12 элементов. Затем заполняет его либо с клавиатуры (разрешается вводить числа в интервале от 0 до 99), либо генерирует его содержимое с помощью [генератора случайных чисел](http://iguania.ru/stati-po-programmirovaniiu/generatsiya-sluchaynich-chisel.html), в интервале от 0 до 99. Для выбора одного из двух вариантов заполнения, программа задает вопрос пользователю. Затем программа должна инвертировать массив, путем замены значения первого элемента на последний, второго на предпоследний и так далее. Вывод на экран должен происходить ровно два раза: до инверсии и после.

Как обычно, напомню, что все задачи на массивы вы должны попытаться выполнить абсолютно самостоятельно. Свое решение в конце каждой задачи я буду приводить лишь для сравнения и помощи в крайних ситуациях, когда после долгих попыток вы так и не сможете найти решение.

Решение задачи

//Задачи на массивы

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

void setRand(int[], const int);

void invertArray(int[], const int);

void printArray(int[], const int);

int main()

{

const int length = 12;

int array[length];

bool label;

srand(time(NULL));

//каким образом будем заполнять массив

cout << "Zapolnenie massiva: sluchajno - 0, rukami - 1 ";

cin >> label;

cout << endl;

//если будем заполнять вручную, вводом с клавиатуры

if(label)

for(int i = 0; i < length; i++)

{

cout << "Vvedite " << i << " ]lement massiva v intervale ot 0 do 99: ";

cin >> array[i];

//если небыл соблюден интервал, то возвращаемся на начало итерации

if(array[i] < 0 || array[i] >= 100)

{

cout << "Vyhod za dopustimyj interval! Povtorite vvod\n";

i--;

}

}

//иначе, заполняем рандомно

else

setRand(array, length);

//заполнили массив - теперь выводим его на экран

//затем инвертируем и опять выводим

printArray(array, length);

invertArray(array, length);

printArray(array, length);

return 0;

}

//рандомное заполнение массива

void setRand(int arr[], const int len)

{

for(int i = 0; i < len; i++)

arr[i] = rand() % 100;

}

//инверсия массива

void invertArray(int arr[], const int len)

{

int temp;

for(int i = 0; i < len / 2; i++)

{

temp = arr[i];

arr[i] = arr[len - i - 1];

arr[len - i - 1] = temp;

}

}

//печать массива

void printArray(int arr[], const int len)

{

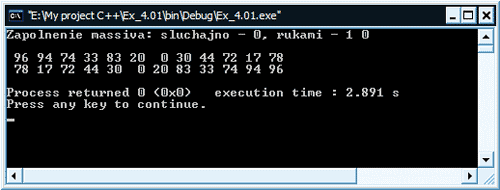
for(int i = 0; i < len; i++)

cout << setw(3) << arr[i];

cout << endl;

}

Результат работы программы



#### Задача 2

В качестве второго задания, предлагаю вам сделать предыдущее задание, но уже использовать не одномерный массив, а двумерный, размером 12 х 12 (матрицу). Единственное, что не нужно делать из предыдущего задания - это предлагать заполнение массива с клавиатуры, т.к. тяжело себе представить ввод пользователем 12 \* 12 = 144 значений. Проделанное задание позволить закрепить навыки работы с двумерными массивами. Как вы, наверное, помните из теории, для прохода по двумерному массиву, удобнее всего использовать два вложенных цикла for.

Решение задачи

//Задачи на массивы

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

void setRand(int[][12], const int);

void invertArray(int[][12], const int);

void printArray(int[][12], const int);

int main()

{

const int length = 12;

int array[length][length];

srand(time(NULL));

setRand(array, length);

printArray(array, length);

invertArray(array, length);

printArray(array, length);

return 0;

}

//рандомное заполнение массива

void setRand(int arr[][12], const int len)

{

for(int i = 0; i < len; i++)

for(int j = 0; j < len; j++)

arr[i][j] = rand() % 100;

}

//инверсия массива

void invertArray(int arr[][12], const int len)

{

int temp;

for(int i = 0; i < len / 2; i++)

for(int j = 0; j < len; j++)

{

temp = arr[i][j];

arr[i][j] = arr[len - i - 1][len - j - 1];

arr[len - i - 1][len - j - 1] = temp;

}

}

//печать массива

void printArray(int arr[][12], const int len)

{

for(int i = 0; i < len; i++)

{

for(int j = 0; j < len; j++)

cout << setw(3) << arr[i][j];

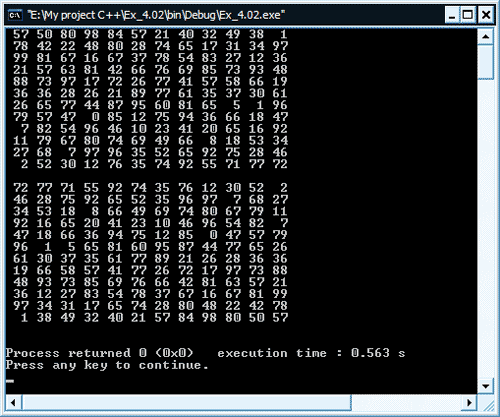
cout << endl;

}

cout << endl;

}

Результат работы программы



#### Задача 3

Если вы успешно справились с двумя предыдущими задачами, то далее решим вот такую несложную задачу на массивы. Создайте двумерный массив, размером 8 х 8. Заполните его случайными величинами в интервале от 20 до 150. Напишите программу, которая находит максимальный и минимальный элементы массива, используя функции minElement() иmaxElement().

Решение задачи

//Задачи на массивы

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

void setRand(int[][8], const int);

void printArray(int[][8], const int);

int minElement(int[][8], const int);

int maxElement(int[][8], const int);

int main()

{

const int length = 8;

int array[length][length];

srand(time(NULL));

setRand(array, length);

printArray(array, length);

cout << "Min element of array = " << minElement(array, length) << endl;

cout << "Max element of array = " << maxElement(array, length) << endl;

return 0;

}

void setRand(int arr[][8], const int len)

{

for(int i = 0; i < len; i++)

for(int j = 0; j < len; j++)

arr[i][j] = 20 + rand() % 130;

}

void printArray(int arr[][8], const int len)

{

for(int i = 0; i < len; i++)

{

for(int j = 0; j < len; j++)

cout << setw(4) << arr[i][j];

cout << endl;

}

cout << endl;

}

int minElement(int arr[][8], const int len)

{

int min = arr[0][0];

for(int i = 0; i < len; i++)

for(int j = 0; j < len; j++)

if(arr[i][j] < min)

min = arr[i][j];

return min;

}

int maxElement(int arr[][8], const int len)

{

int max = arr[0][0];

for(int i = 0; i < len; i++)

for(int j = 0; j < len; j++)

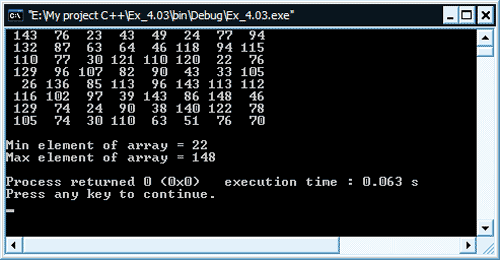
if(arr[i][j] > max)

max = arr[i][j];

return max;

}

Результат работы программы



#### Задача 4

Измените предыдущую программу так, чтобы для нахождения минимального и максимального элементов массива, использовалась лишь одна функция searchElement(). Функция будет принимать в качестве аргументов массив, его размер и логическое значение, указывающее, что нужно найти: min или max.

Решение задачи

//Задачи на массивы

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

void setRand(int[][8], const int);

void printArray(int[][8], const int);

int searchElement(int[][8], const int, const bool);

int main()

{

const int length = 8;

int array[length][length];

srand(time(NULL));

setRand(array, length);

printArray(array, length);

cout << "Min element of array = " << searchElement(array, length, 0) << endl;

cout << "Max element of array = " << searchElement(array, length, 1) << endl;

return 0;

}

void setRand(int arr[][8], const int len)

{

for(int i = 0; i < len; i++)

for(int j = 0; j < len; j++)

arr[i][j] = 20 + rand() % 130;

}

void printArray(int arr[][8], const int len)

{

for(int i = 0; i < len; i++)

{

for(int j = 0; j < len; j++)

cout << setw(4) << arr[i][j];

cout << endl;

}

cout << endl;

}

int searchElement(int arr[][8], const int len, const bool label)

{

int search = arr[0][0];

for(int i = 0; i < len; i++)

for(int j = 0; j < len; j++)

if(label)

{

if(arr[i][j] > search)

search = arr[i][j];

}

else

{

if(arr[i][j] < search)

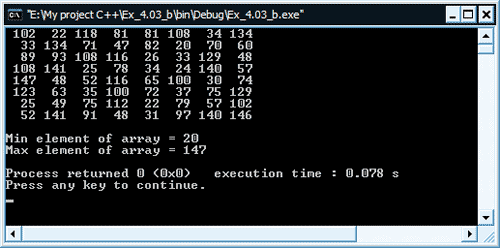
search = arr[i][j];

}

return search;

}

Результат работы программы



На этом решение разминочных задач на массивы закончим, а в следующей главе уже будем решать более сложные и интересные задачи.

#### Задача 5

Данную задачу мы будем решать с использованием одномерного массива. Нужно запросить у пользователя последовательно 20 чисел, каждое из которых должно быть в интервале от 10 до 100 включительно. После того, как программа прочла очередное число, она должна определить: было ли такое число уже введено ранее или нет. Если оно не дублирует ранее прочитанные числа, то напечатайте его, в противном случае не печатайте. Предусмотрите в своей программе "наихудший случай", когда все 20 чисел будут разными. Напишите программу, используя как можно меньшего размера массив.

Ниже привожу в качестве одного из возможных решений, свой вариант решения этой задачи на массив. Ваш вариант реализации, естественно, может быть и не таким, поэтому свой вариант привожу лишь для ознакомления. Настоятельно рекомендую не смотреть и не изучать программу, пока не напишите свою. Это будет большим плюсом для получения вами опыта в решении подобных задач.

//Работа с одномерным массивом

#include <iostream>

using namespace std;

int main()

{

//объявляем необходимые переменные

const int size = 20;

int array[size], var;

bool label = true;

//выводим подсказку пользователю

cout << "Enter 20 numbers in an interval from 10 up to 100" << endl << endl;

//начинаем ввод 20-и чисел в цикле

for(int i = 0; i < size; i++)

{

//запрашиваем число, запоминаем его в переменную var

cout << "Enter: ";

cin >> var;

//если число находится в нужном интервале

if(var >= 10 && var <= 100)

{

//записываем его в массив

array[i] = var;

//проверяем, не дублирует ли оно

//введенное ранее число

for(int j = 0; j < i; j++)

{

//если найдено соответствие, то меняем флаг переменной label

if(array[j] == array[i])

label = false;

}

//если соответствия не было найдено, то печатаем число

if(label == true)

cout << array[i] << endl;

}

//в случае, если было введено число не из интервала 10 - 100

else

{

cout << "Error! Incorrect interval" << endl;

i--;

}

//меняем флаг перед входом в новую итерацию

label = true;

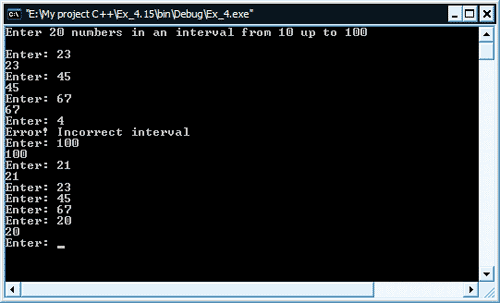
}

return 0;

}

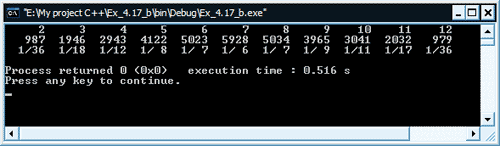
Результат работы программы

В приведенном скрине нет всех двадцати запросов, но видна сама суть работы программы.



#### Задача 6

Следующая задача на массивы будет более интересная. Надеюсь, что вам она тоже покажется именно такой. Нужно написать программу, моделирующую бросание двух костей. Думаю, что все знают, что такое кости. Кто забыл, то напомню, что кости - это шестигранные кубики, каждая грань которых имеет значение очков от 1 до 6 включительно. Соответственно, когда мы бросаем две кости, то количество выпавших очков может быть в интервале от 2 до 12 включительно. По логике вещей и теории вероятностей, количество очков, равное 7, должно выпадать чаще других. Т.к. у обеих костей по шесть значений, то всего может быть 36 различных комбинаций бросков. Их можно рассмотреть на рисунке ниже



Как видите, примерно в 1/6 всех случаев, у нас должно выпадать 7 очков. Вернемся к нашей задачи. Как я сказал выше, нужно написать программу, моделирующую бросание двух костей. Для моделирования бросания обоих костей, программа должна использовать [генератор случайных чисел rand()](http://iguania.ru/tutorial/practice-15-2). Естественно, что поочередно, т.е. один раз получаем рандомное число от 1 до 6 включительно и присваиваем его первой кости, а затем генерируем с помощью rand() второе рандомное число и присваиваем его второй кости. Далее программа подсчитывает сумму их значений. Ваша программа должна бросать две кости ровно 36000 раз. Используйте одномерный массив для подсчета количества раз выпадения той или иной суммы. В конце работы программы выведите результаты в табулированном формате. Определите приемлемость полученных результатов, ориентировочно сравнив их с теоретической вероятностью выпадения того или иного значения.

Мой вариант реализации этого алгоритма

//Моделирование бросания двух костей

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

void printPoints(int pt[], const int len, const int val)

{

for(int i = 0; i < len; i++)

cout << setw(6) << i + 2;

cout << endl;

for(int i = 0; i < len; i++)

cout << setw(6) << pt[i];

cout << endl;

for(int i = 0; i < len; i++)

cout << setw(4) << "1/" << setw(2) << val / pt[i];

cout << endl;

}

int main()

{

const int length = 11, value = 36000;

int points[length] = {0};

int var;

srand(time(NULL));

for(int i = 1; i <= value; i++)

{

var = 1 + rand() % 6;

var += 1 + rand() % 6;

points[var - 2]++;

}

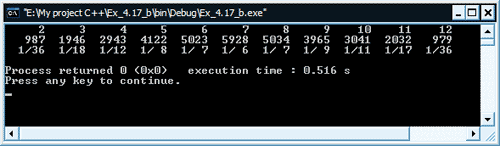
printPoints(points, length, value);

return 0;

}

Результат работы программы

В первом ряду выводятся выпавшие сочетания очков, во втором ряду выводятся количества выпавших сочетаний, а в третьем - общее отношение всех моделирований к выпавшим данного типа. Как видите, для сочетания из 7 очков у нас отношение 1 / 6, значит все нормально работает.



#### Задача 7

Продолжаем решение задач на массивы. Следующая задача, которую я предлагаю вам решить, а вернее написать программу для ее решения, называется "Система резервирования билетов авиакомпании". Вот ее суть:

Небольшая авиакомпания купила компьютеры для реализации своей идеи по резервированию билетов. Вы устроились в эту компанию работать программистом и вас попросили написать программу, которая сможет бронировать места для каждого полета одного единственного самолета. Вместимость этого самолета, работающего на линии этой авиакомпании, составляет 10 мест.

Ваша программа должна отображать следующее меню выбора:

Введите, пожалуйста, 1 для "курящих" Введите, пожалуйста, 2 для "некурящих"

Если клиент ввел 1, то программа должна забронировать для него место в салоне для курящих (это места с 1 по 5 включительно). Если ввел 2, то должна бронировать место в салоне для некурящих (это места с 6 по 10 включительно). Ваша программа должна печатать посадочный талон, в котором будет указываться номер посадочного места и тип салона (для курящих или для некурящих).

Для реализации данной программы используйте одномерный массив, который будет представлять расположение 10 мест самолета. Присвойте всем элементам массива нулевые значения, чтобы показать, что все места свободны. После бронирования места в салоне самолета, выставляйте его в единицу, чтобы показать, что оно уже занято.

Естественно, что ваша программа не должна продавать клиентам уже занятые места. Если салон для курящих уже заполнен, а клиент выбрал первый пункт (салон для "курящих"), то программа должна спросить у клиента "приемлем ли для него салон для некурящих". Если да, то забронируйте для него место, если нет, то программа должна напечатать, что "следующий полет состоится через 3 часа".

Мой вариант решения этой задачи

//Система резервирования билетов авиакомпании

#include <iostream>

#include <iomanip>

using namespace std;

bool checkEmployment(int bk[], const int pl)

{

for(int i = 0; i < pl; i++)

if(bk[i] == 0)

return true;

return false;

}

void printTichet(int place, int salonType)

{

cout << "Vash posadochnyj talon\n\n";

cout << "nomer posadochnogo mesta - " << place << endl;

cout << "tip salona - " << (salonType == 1 ? "dlya kuryaschih\n" : "dlya nekuryaschih\n") << endl;

}

int main()

{

const int places = 10;

int booking[places] = {0};

int salonType, move;

//пока в самолете есть свободные места

while(checkEmployment(booking, places))

{

cout << "Vvedite, pozhalujsta, 1 dlya \"kuryaschih\"" << endl;

cout << "Vvedite, pozhalujsta, 2 dlya \"nekuryaschih\"" << endl;

cin >> salonType;

//если выбран салон для курящих

if(salonType == 1)

{

//ищем свободное место в подходящем салоне

for(int i = 0; i < 5; i++)

{

//если место свободно

if(booking[i] == 0)

{

//помечаем место, как занятое и запоминаем его

booking[i] = 1;

printTichet(i + 1, salonType);

break;

}

//если все места заняты

if(i == 4 && booking[i] == 1)

{

//предлагаем место из салона для некурящих

cout << "Priemlem li dlya vas salon dlya nekuryaschih? (da - 1, net - 0)";

cin >> move;

//если клиент согласен, то присваиваем нужный тип салона

if(move == 1)

salonType = 2;

else

cout << "Sleduyuschij polet sostoitsya cherez 3 chasa\n\n";

}

}

}

//если выбран салон для некурящих

if(salonType == 2)

{

for(int i = 5; i < 10; i++)

{

if(booking[i] == 0)

{

booking[i] = 1;

printTichet(i + 1, salonType);

break;

}

if(i == 9 && booking[i] == 1)

cout << "Sleduyuschij polet sostoitsya cherez 3 chasa\n\n";

}

}

//если было введено неверное число

if(salonType != 1 && salonType != 2)

cout << "Nevernyj vvod! Povtorite vvod esche raz\n";

}

cout << "Vse mesta zanyaty!\n";

return 0;

}

Пожалуй, на сегодня хватит задач на массивы. В этой практической главе, мы решали задачи на одномерные массивы, в следующей будем решать задачи на двумерные массивы.

### Вопрос 1: как заполнить двумерный массив

Да, действительно очень многие новички задают вот такой вопрос: как заполнить двумерный массив. Хочу сказать, что вариантов заполнения массива может быть несколько: это и заполнение массива с клавиатуры, когда пользователь вводит значения, которые необходимо поместить в ячейки массива; это и рандомное заполнение массива, когда в ячейки помещаются псевдослучайные величины, генерируемые с помощью [генератора случайных чисел](http://iguania.ru/tutorial/practice-15-3); это и заполнение массива из файла, в этом случае значения считываются из текстового файла. Начнем с заполнения, путем ввода значений с клавиатуры.

#### Как заполнить двумерный массив с клавиатуры

Довольно часто приходится заполнять массивы с клавиатуры, поэтому подробно рассмотрим этот момент на примере следующей программы

//Как заполнить двумерный массив с клавиатуры

#include <iostream>

#include <iomanip>

using namespace std;

//функция выводит массив на экран

void printArray(int array[][4], const int row, const int column)

{

cout << "\n\nPoluchennyj massiv\n";

for(int i = 0; i < row; i++)

{

for(int j = 0; j < column; j++)

cout << setw(4) << array[i][j];

cout << endl;

}

}

int main()

{

const int row = 3, column = 4; //размеры массива

int array[row][column]; //двумерный массив

for(int i = 0; i < row; i++) //строки массива

{

cout << "Vvodim " << i + 1 << " stroku\n"; //подсказка

for(int j = 0; j < column; j++) //столбцы массива

{

cout << "Vvedite celoe chislo: "; //выводим подсказку

cin >> array[i][j]; //запоминаем введенное значение

}

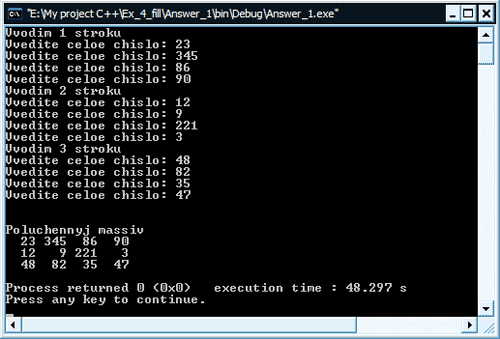
}

printArray(array, row, column); //печатаем полученный массив

return 0;

}

Результат работы программы



Как видите, ничего сложного в заполнении двумерного массива с клавиатуры нет. Мы просто проходим двумя циклами **for**(один вложен в другой) по массиву. Первый цикл считает строки, второй столбцы. Когда все столбцы (значения) первой строки пройдены, то основной цикл переходит на вторую строку, а вложенный цикл опять начинает проходить по строке с начала, считая ее столбцы.

P.S. Не устаю напоминать, что нумерация элементов (ячеек) массива **начинается с нуля**. Следите за этим, чтобы не было ошибок.

#### Как заполнить двумерный массив случайными числами

Иногда приходится рандомно заполнять массив. На примере следующей программы мы разберем то, как заполнять двумерный массив случайными числами. Смотрим код

//Как заполнить двумерный массив случайными числами

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <time.h>

using namespace std;

//функция выводит массив на экран

void printArray(int array[][4], const int row, const int column)

{

cout << "Poluchennyj massiv\n";

for(int i = 0; i < row; i++)

{

for(int j = 0; j < column; j++)

cout << setw(4) << array[i][j];

cout << endl;

}

}

int main()

{

const int row = 5, column = 4; //размеры массива

int array[row][column]; //двумерный массив

srand(time(NULL));

for(int i = 0; i < row; i++) //строки массива

for(int j = 0; j < column; j++) //столбцы массива

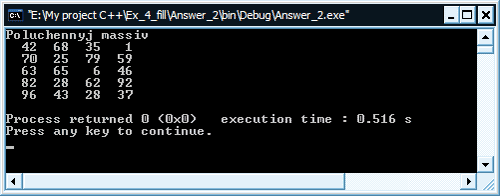
array[i][j] = 1 + rand() % 100; //заполняем текущую ячейку

printArray(array, row, column); //печатаем полученный массив

return 0;

}

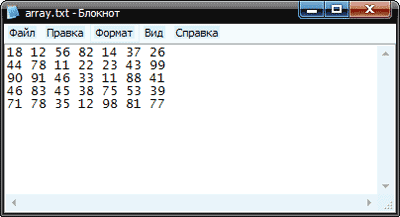
Результат работы программы



Для генерации случайных величин в С++ служит функция **rand()**. В данном случае она генерирует псевдослучайные величины в интервале от 1 до 100, т.к. началом у нас задана величина 1, а смещением величина 100. Подробно это написано в статье о [функции rand()](http://iguania.ru/tutorial/practice-15-3).

#### Как заполнить двумерный массив из файла

Рассмотрим последний вариант заполнения - это заполнение двумерного массива из файла. Естественно, что в папке с проектом у нас должен быть заранее подготовленный текстовый файл array.txt, с которого мы будем читать числа в массив. У меня был подготовлен файл следующего содержания



Кстати, на форуме есть тема, посвященная [чтению и записи из файлов](http://iguania.ru/tutorial/practice-15-3), где можно подробно почитать о чтении из файла и записи в файл. Смотрим код

//Как заполнить двумерный массив из файла

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

//функция выводит массив на экран

void printArray(int array[][7], const int row, const int column)

{

cout << "Poluchennyj massiv\n";

for(int i = 0; i < row; i++)

{

for(int j = 0; j < column; j++)

cout << setw(4) << array[i][j];

cout << endl;

}

}

int main()

{

const int row = 5, column = 7;

int array[row][column];

//создаем объект класса ifstream

//и сразу указываем файл, с которого будем читать

ifstream f("array.txt");

//читаем файл и прочитанное запоминаем в массив

for(int i = 0; i < row; i++)

for(int j = 0; j < column; j++)

f >> array[i][j];

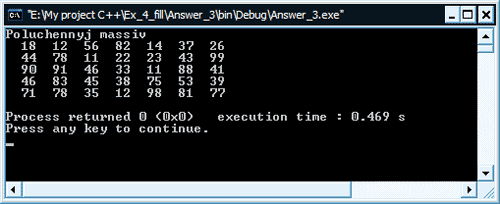
//печатаем полученный заполненный массив

printArray(array, row, column);

return 0;

}

Результат работы программы



Как видите, стандартное чтение из файла выполняется достаточно просто. Для этого мы используем класс **ifstream**, на основе которого создаем объект **f** и указываем файл, с которого будем выполнять чтение. Далее в циклах (они нужны, как вы знаете, для похода по двумерному массиву и, соответственно, по файлу) мы с помощью уже привычного оператора ( >> ) "взять из потока", берем из потока (в потоке у нас находится очередное считанное из файла значение) значение и сохраняем его в очередную ячейку массива.