Генератор случайных чисел. Использование ГСЧ для заполнения массива

В языке С существует возможность генерировать случайное число. Для этой операции используется функция под названием **rand(). Эта функция находиться в** библиотечном файле — stdlib.h, следовательно для ее работы необходимо этот файл подключить с помощью директивы #include. На место вызова **rand() в программе,** подставится случайное число в диапазоне от 0 до RAND_MAX. RAND_MAX это константа, определенная в <cstdlib> (32767)

```
#include"stdafx.h"
#include<iostream>
#include<stdlib.h>// в этом файле содержится функция rand

using namespace std;

void main()

{

int a;
//генерация случайного числа и запись его в переменную a
a = rand();

cout << a << "\n";
/* повторная генерация случайного числа и запись его в переменную a */
a = rand();

cout << a << "\n";
cout << "RAND_MAX = " << RAND_MAX << endl;
}
```

Вывод :

41 и 18467 Эти значения будут повторяться !

Функция **rand()** работает, используя в качестве начальной

точки — точку определенную при написании алгоритма

генератора случайного числа, то есть

некое постоянное число. Другими словами, опираясь на

эту точку, при разных вызовах программы эта функция

генерирует одно и то же число, в чём мы уже успели убе-

диться. Для того, чтобы rand() при разных вызовах

программы выдавал разные числа необходимо изменить

начальную точку генерации

- **Использование функции srand**

Функция **srand устанавливает начальную точку для** генерации случайных чисел и обладает следующим синтаксисом:

void srand(unsigned int start)

```
#include"stdafx.h"
#include<iostream>
//в этом файле содержатся функции rand и srand
#include<stdlib.h>
using namespace std;
void main()
{
int start=25;
srand(start);
int a;
//генерация случайного числа и запись его в переменную
a=rand();
cout<<a<<"\n";
}
```

Использование функции time

Местоположение функции — библиотека time.h.

Если функцию time вызвать с параметром NULL, то на место

своего вызова в программе, эта функция вернет количество

миллисекунд прошедших с 1 января 1970 года.

Эта величина каждый раз будет разной.

srand(time(NULL));

Функция srand устанавливает в качестве стартовой точки

число, представляющее собой количество, миллисекунд

прошедших с 1 января 1970 года.

```
#include"stdafx.h"
#include<iostream>
#include<stdlib.h>// в этом файле содержатся rand и srand
#include<time.h>// в этом файле содержится функция time
using namespace std;
void main()
{
srand(time(NULL));
int a;
//генерация случайного числа и запись его в переменную
a=rand();
cout<<a<<"\n";
}
```

Числа, которые получаются путем вызова функции rand, находятся в диапазоне от 0 до 32767.

Правило №1

ЧИСЛО В ДИАПАЗОНЕ ОТ НУЛЯ ДО X:

rand() % X

Пример с числом 23

Какое бы число вы не разделили на 23 по модулю, вы получите

либо 0 (если остатка нет), либо остаток в диапазоне от 1 до 22.

Т. к. диапазон не всегда начинается с нуля. Пусть нам необходим диапазон от 11 до 16. Все просто. Необходимо генерировать числа от 0 до 5 (разница между 16 и 11), а потом «сдвинуть» полученный результат на 11 единиц.

Правило №2

ЧИСЛО В ДИАПАЗОНЕ ОТ Y ДО X:

rand() % (X-Y) +Y

rand() % 10 + 1 ?

- Пример использования rand для заполнения массива

```
#include"stdafx.h"
#include<iostream>
#include<stdlib.h>// в этом файле содержатся rand и srand
#include<time.h>// в этом файле содержится функция time
using namespace std;
void main()
{
srand(time(NULL));
int array[10];
for (int i = 0; i<10; i++)
{
// генерация случайного числа и запись его в текущий элемент
    массива
array[i] = rand() % 100;
// показ значения элемента на экран
cout << array[i] << endl;
}
}
```

- Двумерный массив.

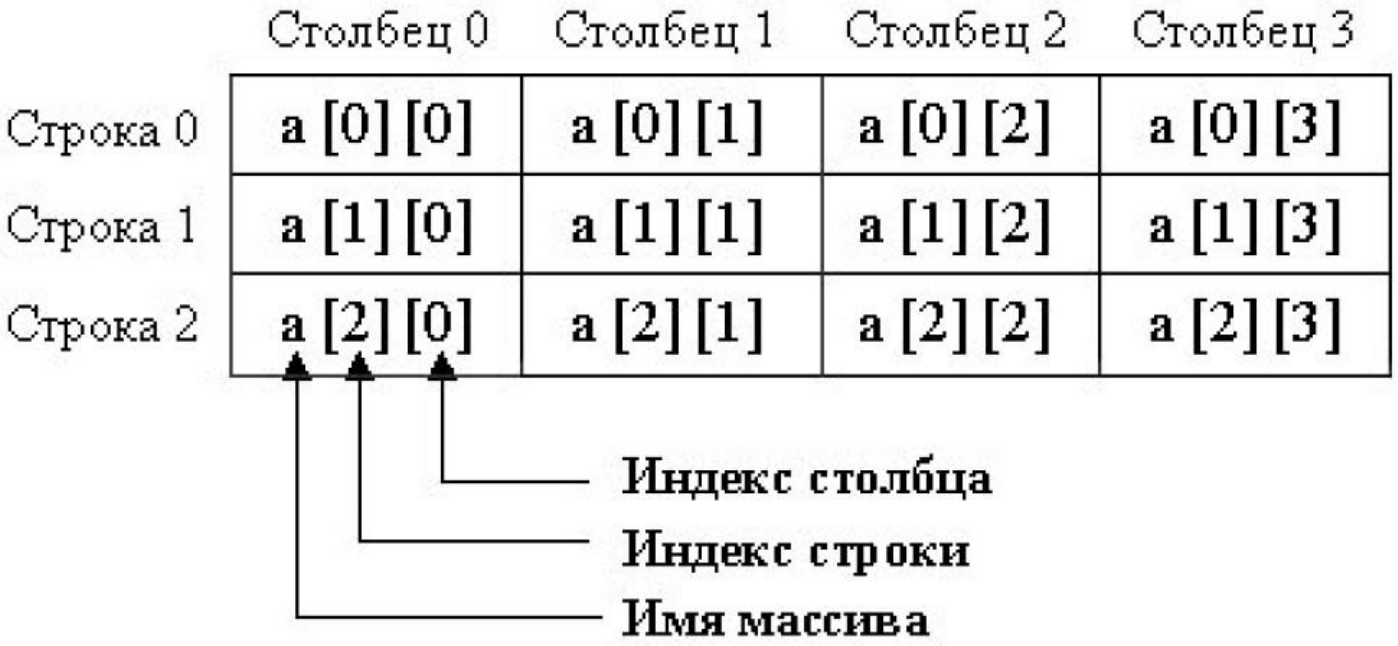
Двумерный массив представляет собой совокупность строк и
столбцов, на пересечении которых находится конкретное
значение. Объявить двумерный массив несложно, необходимо
указать количество строк и столбцов.

Общий синтаксис:

тип_данных имя_массива [число_строк][число_столбцов];

Пример:

```
const int row=3; // строки
const int col=4; // столбцы
int array[row][col]; // массив размером row на col(3x4)
```


Столбец 0
Столбец 1
Столбец 2
Столбец 3
Строка 0
Строка 1
Строка 2
a [0] [0]
a [0] [1]
a [0] [2]
a [0] [3]
a [1] [0]
a [1] [1]
a [1] [2]
a [1] [3]
a [2] [0]
a [2] [1]
a [2] [2]
a [2] [3]
Индекс столбца
Индекс строки
Имя массива

Несмотря на то, что мы представляем массив в

виде матрицы, на самом деле — любой

двумерный массив располагается в памяти

построчно: сначала нулевая строка, затем

первая и так далее. Об этом следует помнить,

т.к. выход за пределы массива может повлечь за

1. Инициализация при создании.

Каждая строка заключается в отдельные фигурные скобки:

- int array[2][2]={{1,2},{7,8}};

Значения указываются подряд и построчно вписываются в массив:

- int array[2][2]={7,8,10,3};

Если значение пропущено, оно будет инициализировано нулем:

- int array[3][3]={{7,8},{10,3,5}};

Заполнение двумерного массива случ. значениями

```
#include"stdafx.h"
#include<iostream>
#include<stdlib.h> //в этом файле содержатся rand и srand
#include<time.h> // в этом файле содержится функция time
using namespace std;
void main()
{
const int row = 3; // строки
const int col = 3; // столбцы
int mr[row][col]; // массив размером row на col
/* перебираем отдельные строки (одномерные массивы в совокупнотси) */
for (int i = 0; i<row; i++)
{
// перебираем отдельные элементы каждой строки
for (int j = 0; j<col; j++)
{

mr[i][j]=rand()%100;
// показ значений на экран
cout<<mr[i][j]<<" ";
}
// переход на другую строку матрицы
cout<<"\n\n";
}
}
```

Поиск max значения в каждой строке

```
#include"stdafx.h"
#include<iostream>
#include<stdlib.h> //в этом файле содержатся rand и srand
#include<time.h> // в этом файле содержится функция time
using namespace std;
void main()
{
// задаем размерность массива
const int m = 3;
const int n = 2;
int A[m][n]; // объявляем двумерный массив
// заполнение массива случайными числами и показ на экран
// перебираем отдельные строки
setlocale(LC_ALL, "rus");
cout.width(2);
for (int i = 0; i<m; i++)
{
// перебираем отдельные элементы каждой строки
for (int j = 0; j<n; j++)
{
// инициализация элементов значениями в диапазоне от 0 до 100
A[i][j] = rand() % 100;
// показ значений на экран
cout << A[i][j] << " ";
}
// переход на другую строку матрицы
cout << "\n\n";
}
cout << "\n\n";
// поиск в строках максимального элемента
// перебираем отдельные строки
for (int i = 0; i<m; i++)  {

// предполагаем, что максимальный - нулевой элемент строки
int max = A[i][0];
// поиск максимального элемента в текущей строке
// изменение индекса столбца для текущей строки
for (int j = 0; j<n; j++)
{
if (A[i][j] > max)
max = A[i][j];
}
cout << "Максимальный элемент " << i
<< "-ой строки = " << max << endl;
}
}
```

Алгоритм генерации случайных УНИКАЛЬНЫХ чисел

```
#include"stdafx.h"
#include<iostream>
#include<stdlib.h> //в этом файле содержатся rand и srand
#include<time.h> // в этом файле содержится функция time
using namespace std;
void main()
{
const int AMOUNT=10; //amount of random numbers that need to be generated
const int MAX=10; //maximum value (of course, this must be at least the same as AMOUNT;

int value[AMOUNT]; //array to store the random numbers in

srand(time(NULL)); //always seed your RNG before using it

//generate random numbers:
for (int i = 0; i<AMOUNT; i++)
{
bool check; //variable to check or number is already used
int n; //variable to store the number in
do
{
n = rand() % MAX;
//check or number is already used:
check = true;
for (int j = 0; j<i; j++)
if (n == value[j]) //if number is already used
{
check = false; //set check to false
break; //no need to check the other elements of value[]
}
} while (!check); //loop until new, unique number is found
value[i] = n; //store the generated number in the array
}

//at this point in the program we have an array value[] with a serie of unique random numbers

for (int i = 0; i < AMOUNT; i++)
{
cout << value[i] << " ";

}
}
```

```
#include "stdafx.h"
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(){
//declaring array
const  int N=5;
int array[N];
cout << "Enter 5 numbers randomly : " << endl;
for (int i = 0; i<N; i++)
{
//Taking input in array
cin >> array[i];
}
cout << endl;
cout << "Input array is: " << endl;

for (int j = 0; j<N; j++)
{
//Displaying Array
cout << "\t\t\tValue at " << j << " Index: " << array[j] << endl;
}
cout << endl;
// Bubble Sort Starts Here
int temp;
for (int i2 = 0; i2 <= N-1; i2++)
{
for (int j = 0; j<N-1; j++)
{
//Swapping element in if statement
if (array[j]>array[j + 1])
{
temp = array[j];
array[j] = array[j + 1];
array[j + 1] = temp;
}
}
}
// Displaying Sorted array
cout << "  Sorted Array is: " << endl;
for (int i3 = 0; i3<N; i3++)
{
cout << "\t\t\tValue at " << i3 << " Index: " << array[i3] << endl;
}
return 0;
}
```

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N − 1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу

B	20	10	8	7	5	2
1-й шаг	10	8	7	5	2	20
2-й шаг	8	7	5	2	10	20
3-й шаг	7	5	2	8	10	20
4-й шаг	5	2	7	8	10	20
5-й шаг	2	5	7	8	10	20