МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования "**Национальный исследовательский** **Нижегородский** **государственный** **университет** **им**. **Н.И. Лобачевского**" (**ННГУ**)

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Отчет по Лабораторной работе №1

**Генерация случайных чисел в заданном диапазоне и работа с ними**

Выполнил:

студент группы 3821Б1ПМ3

Логинов В.А.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород  
2021

Оглавление

[1. Введение 3](#_Toc86278975)

[2. Постановка задачи 4](#_Toc86278976)

[3. Руководство пользователя 5](#_Toc86278977)

[4. Руководство программиста 6](#_Toc86278978)

[4.1. Описание структуры программы 6](#_Toc86278979)

[4.2. Описание структур данных 9](#_Toc86278980)

[4.3. Описание алгоритмов 10](#_Toc86278981)

[5. Экперементы 13](#_Toc86278982)

[6. Заключение 14](#_Toc86278983)

[7. Литература 15](#_Toc86278984)

[8. Приложение 16](#_Toc86278985)

# Введение

Любому начинающему программисту необходимо освоить базовые принципы работы языков программирования. Язык С нередко называют языком программирования "среднего уровня" или даже "низкого уровня", так как он сочетает элементы языков высокого уровня с функциональностью и производительностью ассемблера и работает близко к аппаратной части компьютера. В итоге мы можем манипулировать данными на низком уровне и при этом использовать высокоуровневые конструкции для управления работы программы. Поэтому он прекрасно подходить для изучения работы аппаратной части компьютера.

Неотъемлемой частью работы компьютера является использование RAM. Понимание того, на что и чем выделяется память необходимо для написания эффективной в работе программы. В частности, использование динамических массивов в данной работе требовало умения непосредственной работы с оперативной памятью.

Также, в большинстве программ по анализу данных, происходить работа с числами, поэтому необходимо разбираться в типах данных и их применении, для симуляции работы с настоящими данными была использована случайная генерация чисел.

# Постановка задачи

Программа генерирует множество случайных чисел, их количество(N), нижнюю(niz) и верхнюю(verx) границу пользователь вводит с клавиатуры. После этого программа подсчитывает и выводит сумму, формируемую следующим образом: числа, порядковые номера которых совпадают с дробной частью хотя бы одного из вводимых чисел – вычитаются, все остальные – прибавляются.

# Руководство пользователя

После запуска программы, откроется окно командной строки, где вас попросят последовательно ввести количество генерируемых программой чисел (рис.1), нижнюю (рис.2) и верхнюю границу (рис.3) диапазона генерации числа. После ввода каждого параметра необходимо нажимать клавишу «Enter».



Рис.1 Окно после запуска программы

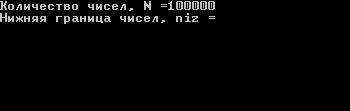


Рис.2 Окно после ввода количества чисел

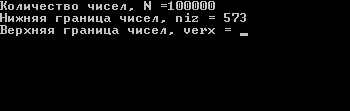


Рис.3 Окно после ввода нижней границы

Если по введенным данным невозможно исполнить программу, то вы будете уведомлены об этом следующим сообщением:

лр3.png

Рис.4 Уведомление о некорректных данных

Если требуемое вами количества чисел будет больше 250 000 000, то исполнение программы будет остановлено и вы увидите сообщение:

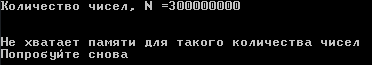


Рис.6 Уведомление о слишком большом запрашиваемом количестве генерируемых чисел

# Руководство программиста

## Описание структуры программы

1. В программе присутствует множество модулей. Первые строчки стандартные для большинства программ, здесь подключаются необходимые библиотеки.

##### Фрагмент кода 1. Подключение библиотек

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <locale.h>

1. Далее происходит объявлении двух функций, первая для генерации случайного числа в заданном диапазоне, вторая для получения дробной части от числа с точностью до некоторого знака. Их работа подробно будет изложена в разделе «Описание алгоритмов».

##### Фрагмент кода 2. Объявление функций

double dr\_chislo\_diap(int aa, int bb)

{

double a;

double b = 0.0;

a = rand();

b = (a / RAND\_MAX) \* (bb-aa)+aa;

return b;

}

int dr\_chast(double aa)

{

int m1 = 0, a1 = 0;

double a = 0.0;

a = aa;

a1 = (int)a;

a = a - a1;

a = a \* 1000000;

m1 = (int)a;

return m1;

}

1. После этого идет объявление с инициализацией всех необходимых переменны, а также подключение кодировки для русского алфавита, для корректного отображения выводимых далее на экран сообщений:

##### Фрагмент кода 3. Объявление и инициализация переменных

setlocale(LC\_ALL, "Russian");

int i = 0, j = 0, k = 0 ;

int niz = 0, verx = 0;

int dr\_ch\_c = 0;

int N = 0;

double\* mas;

double\* mas2;

double summa = 0.0;

double ch = 0.0;

1. Следующий блок отвечает за заполнение необходимых переменных данными, получаемыми от пользователя при помощи команды «scanf\_s», при этом указывая за что будет отвечать вводимое значение выводя на экран текстовые пояснения при помощи команды «printf».

##### Фрагмент кода 4. Заполнение переменных данными пользователя

printf("Количество чисел, N =");

scanf\_s("%d", &N);

printf("Нижняя граница чисел, niz = ");

scanf\_s("%d", &niz);

printf("Верхняя граница чисел, verx = ");

scanf\_s("%d", &verx);

1. Далее идут условные операторы if проверяющие на корректность все введенный данные, в случае не корректности какой либо переменной, программа выводит сообщение об этом и прекращает свое исполнение. Отдельно проверяется введенный размер массива, чтобы в случае слишком большого размера не произошло резервирование всей RAM и зависания компьютера.

##### Фрагмент кода 5. Проверка введенных данных

if (N > 250000000)

{

printf("\n\nНе хватает памяти для такого количества чисел\nПопробуйте снова\n\n");

return 1;

}

if (N > 250000000)

{

printf("\n\nНе хватает памяти для такого количества чисел\nПопробуйте снова\n\n");

return 1;

}

1. В следующих двух строчках заканчивается объявление переменных, объявлением 2-х массивов, основного и вспомогательного:

##### Фрагмент кода 6. Дообъявление переменных.

mas = (double\*)malloc(N \* sizeof(double));

1. Следующий блок кода это цикл for который заполняет оба массива данными получаемыми вызовом функции dr\_chislo\_diap:

##### Фрагмент кода 7. Заполнение массивов

for (i = 0; i < N; i++)

{

double a = dr\_chislo\_diap(niz, verx);

mas[i] = a;

mas2[i] = a;

}

1. Далее идет основной модуль, отвечающий за получение итоговой суммы последовательным исполнением двух циклов for, вывод итоговой суммы на экран и очистку памяти от массивов (подробно рассматривается в «Описание алгоритмов»):

##### Фрагмент кода 8.

for (j = 0; j < N; j++)

{

ch = mas[j];

dr\_ch\_c = dr\_chast(ch);

if (dr\_ch\_c < N)

{

summa -= mas[dr\_ch\_c];

mas[dr\_ch\_c] = 0.0;

}

}

for (k = 0; k < N; k++)

{

if (mas[k] != 0.0)

summa += mas[k];

}

printf("\n\n\nСумма = %lf\n\n\n", summa);

free(mas);

return 0;

## Описание структур данных

* + 1. В программе использованы несколько типов данных:
       1. Для счетчиков в циклах i, j, k использован int так как счетчики обязаны быть целочисленными. Также int использован для N- количества чисел в массиве, так как оно не может быть дробным и для dr\_ch\_c – дробной части числа зависанной целым числом, так как для работы с ней необходим целочисленный формат.
       2. Верхняя и нижняя граница диапазона генерируемых чисел имеет тип double так как может потребоваться не целочисленная граница. Подсчитываемая сумма (summa) и переменная ch(отвечает за временное сохранение числа из массива) так же имеют тип double так как программа работает с не целыми числами.
       3. 2 массива имеют тип данных double, так как хранят дробные числа.
    2. Также использованы следующие библиотеки:
       1. #include <stdio.h> - стандартный заголовочный файл ввода-вывода
       2. #include <stdlib.h> - для работы с массивами
       3. #include <locale.h> - для корректного отображения сообщений на русском языке

## Описание алгоритмов

В программе реализованы 3 алгоритма: генерация случайного числа в заданном диапазоне, получение дробной части числа в виде целого(первые 5 знаков),подсчет суммы с запоминание не использованных в подсчете чисел.

1. Генерация случайного числа в заданном диапазоне.

За генерация случайного числа отвечает функция dr\_chislo\_diap принимающая на вход 2 числа, нижнюю и верхнюю границу требуемого диапазона соответственно. После генерируется псевдослучайное целое число встроенной функцией rand и присваивается переменной a. Для получения дробного числа, a делится на константу RAND\_MAX. А для получение числа в диапазоне, предыдущий результат умножается на разность верхней и нижней границы и суммируется с нижней границей. Функция возвращает число b- случайное дробное число в заданном диапазоне.

Фрагмент кода 9. Генерация случайного числа

double dr\_chislo\_diap(double niz, double verx)

{

double a;

double b = 0.0;

a = rand();

b = (a / RAND\_MAX) \* (verx-niz)+niz;

return b;

1. Получение дробной части.

Функция dr\_chast принимает на вход дробное число, получает его целою часть явным приведение типов сохраняя ее в переменную a1, далее вычитает из дробного числа его целую часть, получая таким образом число с нулевой целой частью, но сохраненной дробной, Умножением на 100 000 и последующие явное приведение к типу int позволяют получить дробную часть точностью 5 знаков. Функция возвращает первые 5 знаков дробной части исходного числа.

##### Фрагмент кода 10. Получение дробной части.

int dr\_chast(double aa)

{

int m1 = 0, a1 = 0;

double a = 0.0;

a = aa;

a1 = (int)a;

a = a - a1;

a = a \* 1000000;

m1 = (int)a;

return m1;

}

1. Подсчет итоговой суммы.

Для того чтобы подсчитать сумму исполняются 2 цикла for. Первый пробегает по всем элементам массива, для каждого получает дробную часть функцией dr\_chast , и если дробная часть может являться адресом в массиве(она <= размера основного массива), то из суммы вычитается число, адрес которого равен полученной дробной части. После в массиве обнуляет элемент, который только что был вычтен из суммы. Второй цикл завершает получение ответа, пробегаясь по всем элемента массива и прибавляя к сумме все не нулевые элементы.

Фрагмент кода 11. Подсчет итоговой суммы.

for (j = 0; j < N; j++)

{

ch = mas[j];

dr\_ch\_c = dr\_chast(ch);

if (dr\_ch\_c < N)

{

summa -= mas[dr\_ch\_c];

mas[dr\_ch\_c] = 0.0;

}

}

for (k = 0; k < N; k++)

{

if (mas[k] != 0.0)

summa += mas[k];

}

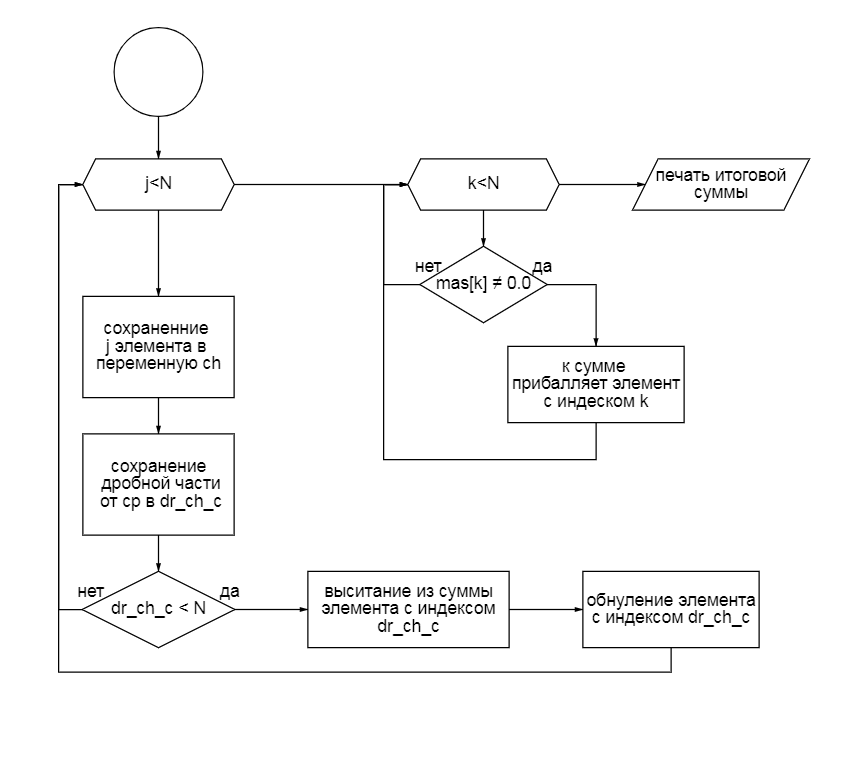


Рис.7 Блок-схема алгоритма подсчета суммы по тех. заданию

## Эксперименты

После написания программы необходимо провести пробные запуски с различными условия, чтобы убедится в правильности работы. Для начала введем верные данные и проверим общую работоспособность, как видно, все работает:

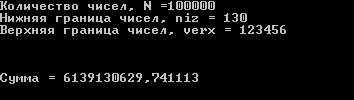


Рис.8 Результат при верных входных данных

Так как в программе есть 2 условия на не корректные данные, то проверим работы программы с данными для срабатывания по каждому из условий. Видно, что программа успешно сообщает пользователю о не корректных данных, как по размеру создаваемого массива(рис 9), так и по не верным границам(рис 10):

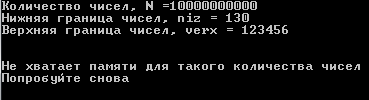


Рис.9 Уведомление при слишком большом количестве чисел

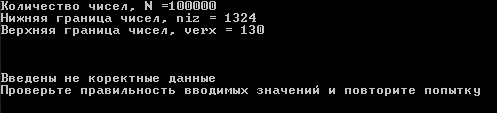


Рис.10 Уведомление при введении неверных границ генерируемых чисел

Введем теперь небольшие корректные данные и проверим правильность подсчета суммы, как видно (Рис.11), сумма подсчитана верно:

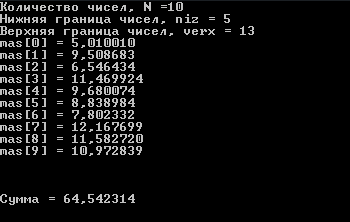


Рис.11 Правильна сумма для данных входных значений

# Заключение

В ходе лабораторной работы была написана программа на языке программирования С, которая успешно справляется с поставленной задачей, а именно: Программа генерирует множество случайных чисел, их количество(N), нижнюю(niz) и верхнюю(verx) границу пользователь вводит с клавиатуры. После этого программа подсчитывает и выводит сумму, формируемую следующим образом: числа, порядковые номера которых совпадают с дробной частью хотя бы одного из вводимых чисел – вычитаются, все остальные – прибавляются.

Работа программы интуитивна понятна, все необходимые инструкции в ходе использования программы пользователь получает на русском языке, в кратком, но содержательном виде.

Для выполнения лабораторной работы были изучены методы работы со случайными числами, способы получения дробной части числа, а также система работы с динамическими массивами.

# Литература

* + 1. Динамическое выделение памяти в Си // Програмирование С и С++ – Режим доступа: <https://prog-cpp.ru/c-alloc/>
    2. [OldNileCrocodile](https://habr.com/ru/users/OldNileCrocodile/) Кратко об указателях в Си: присваивание, разыменование и перемещение по массивам // Хабр – Режим доступа: <https://habr.com/ru/post/545674/>
    3. Пользовательские функции в Си // Young Coder – Режим доступа: https://youngcoder.ru/lessons/10/polzovatelskie\_funkcii.php

# Приложение

#include <stdio.h>

#include <stdlib.h>

#include<locale.h>

double dr\_chislo\_diap(double niz, double verx)

{

double a;

double b = 0.0;

a = rand();

b = (a / RAND\_MAX) \* (verx-niz)+niz;

return b;

}

int dr\_chast(double aa)

{

int m1 = 0, a1 = 0;

double a = 0.0;

a = aa;

a1 = (int)a;

a = a - a1;

a = a \* 1000000;

m1 = (int)a;

return m1;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i = 0, j = 0, k = 0 ;

int dr\_ch\_c = 0;

int N = 0;

double niz = 0.0, verx = 0.0;

double summa = 0.0;

double ch = 0.0;

double\* mas;

double\* mas2;

printf("Количество чисел, N =");

scanf\_s("%d", &N);

printf("Нижняя граница чисел, niz = ");

scanf\_s("%lf", &niz);

printf("Верхняя граница чисел, verx = ");

scanf\_s("%lf", &verx);

if (N > 250000000)

{

printf("\n\nНе хватает памяти для такого количества чисел\nПопробуйте снова\n\n");

return 1;

}

if(((verx > niz) && (niz >= 0) && (N >= 1))== 0)

{

printf("\n\n\nВведены не коректные данные\nПроверьте правильность вводимых значений и повторите попытку\n\n\n");

return 0;

}

mas = (double\*)malloc(N \* sizeof(double));

for (i = 0; i < N; i++)

{

double a = dr\_chislo\_diap(niz, verx);

mas[i] = a;

mas2[i] = a;

}

for (j = 0; j < N; j++)

{

ch = mas[j];

dr\_ch\_c = dr\_chast(ch);

if (dr\_ch\_c < N)

{

summa -= mas[dr\_ch\_c];

mas[dr\_ch\_c] = 0.0;

}

}

for (k = 0; k < N; k++)

{

if (mas[k] != 0.0)

summa += mas[k];

}

printf("\n\n\nСумма = %lf\n\n\n", summa);

free(mas);

return 0;

}