

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования “Нижегородский Государственный Университет им. Н. И. Лобачевского”

Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе

Генерация случайных чисел в заданном диапазоне и работа с ними

Выполнил:
Студент группы 3821Б1ПМЗ

Колганов Д. В.

Проверил:

Заведующий лабораторией
суперкомпьютерных
технологий и
высокопроизводительных
вычислений

Лебедев И.Г

Нижний Новгород

2021 г.

Содержание

Введение	3
Постановка задачи	4
Руководство пользователя.....	5
Руководство программиста	6
<i>Описание структуры программы</i>	6
<i>Описание структур данных</i>	7
<i>Описание алгоритмов</i>	8
Эксперименты	10
Заключение	12
Литература.....	13
Приложение	14

Введение

В настоящее время программирование – одна из основополагающих отраслей в мироустройстве. На работе программистов завязано многое – множественные расчеты (от экономической сферы, до космической), создание необходимого программного обеспечения, инструментов развлечения, и много другого.

Множество работ, выполненное на данный момент, поражает воображение, но для создания столь трудных программ стоит изначально изучить основы программирования, одной из которых и являются случайные числа, а также работа с ними.

Случайные числа используются повсеместно: от криптографических систем шифрования до генерации индивидуальных идентификаторов.

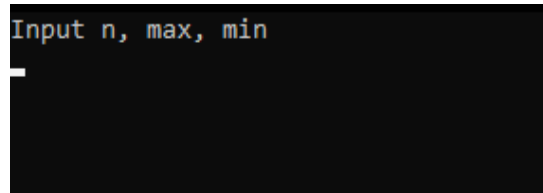
Данная лабораторная работа направлена на изучение генерации случайных чисел, работы с ними на базе языка программирования “С”.

Постановка задачи

Программа на вход получает количество случайных элементов, а также значения максимального и минимального элементов. В результате работы программы на экран должно быть выведено число, представляющее собой сумму элементов, которая считается по данному алгоритму: все числа, номера которых совпадают с дробной частью одного из исходных чисел – вычитаются, остальные – складываются.

Руководство пользователя

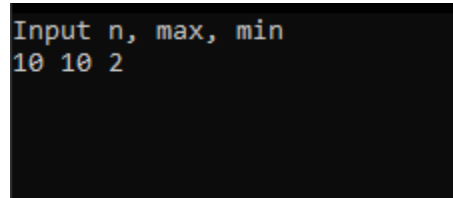
При запуске программы пользователю в консоли выводится сообщение: “Input n, max, min” (см. Рис. 1.1)



```
Input n, max, min
_
```

Рисунок 1.1 – консоль после запуска программы.

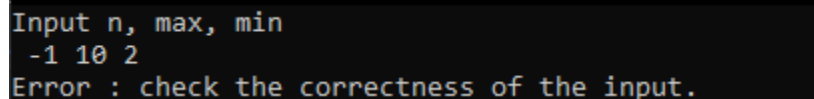
Далее пользователю нужно ввести с клавиатуры значения, соответствующие заданной переменной (n, max, min). Значения нужно вводить в порядке расположения переменных в консоли. (см. рис. 1.2)



```
Input n, max, min
10 10 2
```

Рисунок 1.2 – консоль после ввода данных.

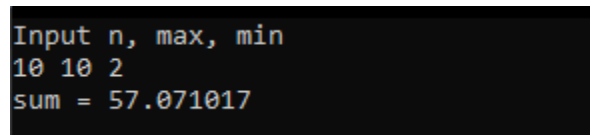
После ввода пользователем данных, программа проверяет корректность ввода и в случае некорректного ввода выводит на экран консоли ошибку: “Error : check the correctness of the input.”. После чего пользователю стоит удостовериться в правильности ввода данных (($n > 0$) и max не должен быть меньше или равен min) (см. рис. 1.3)



```
Input n, max, min
-1 10 2
Error : check the correctness of the input.
```

Рисунок 1.3 – Сообщение об ошибке ввода.

При корректном вводе данных пользователю показывается сообщение “sum = ... “ и соответственно само значение полученной суммы. (см. рис. 4.4)



```
Input n, max, min
10 10 2
sum = 57.071017
```

Рисунок 1.4 – Результат работы программы.

Руководство программиста

Описание структуры программы

1. Подключение необходимых для работы программы библиотек.(см. рис. 2.1)

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
```

Рисунок 2.1. – Подключение библиотек.

2. Объявление функции int main(), блок объявления переменных. (см. рис. 2.2)

```
int main()
{
    int n = 0, i = 0;
    double min = 0.0, max = 0.0, sum = 0.0, rem = 0.0;
```

Рисунок 2.2 – Объявление блока main и переменных.

3. Запрос ввода исходных данных, последующее их сканирование и проверка на корректность ввода (соответственно с помощью команд printf, scanf_s, и оператора if). (см. рис. 2.3)

```
printf("Input n, max, min\n");
scanf_s("%d %lf %lf", &n, &max, &min);

if ((n <= 0) || (max <= min)) printf("Error : check the correctness of the input.");
else
```

Рисунок 2.3 – Запрос, сканирование и проверка данных.

4. Далее расположен блок кода, в котором инициализируются 2 динамических массива размера n (получено в предыдущем блоке): masOsn и masRem. masOsn хранит случайные числа в заданном пользователем диапазоне, а masRem – дробные части исходных случайных чисел. (Заполнение массива masOsn происходит с помощью цикла “for”, а выделение дробной части – с помощью цикла “while”) (см. рис. 2.4)

```
double* masOsn = (double*)malloc(sizeof(double) * n);
int* masRem = (int*)malloc(sizeof(int) * n);

for (i = 0; i < n; i++)
{
    masOsn[i] = (((double)rand() / RAND_MAX) * (max - min) + min);
    rem = ((masOsn[i] - (int)masOsn[i]) * 10000);
    if (rem != 0)
    {
        masRem[i] = (int)rem;
        while (masRem[i] % 10 == 0)
            masRem[i] = masRem[i] / 10;
```

Рисунок 2.4. – Работа с массивами и приведение остатков

5. Последний блок программы отвечает за суммирование и вычитание конкретных элементов массива, отвечающих заданным условиям поставленной задачи. С помощью цикла “for” мы, сначала, начинаем проверять оператором “if” дробные части чисел, приведенные к заданному виду. После, при совпадении из итоговой суммы мы вычитаем элемент от данного номера, в ином случае – прибавляем его. (см. рис. 5.5)

```
for (i = 0; i < n; i++)
{
    if (masRem[i] <= n)
        sum = sum - masOsn[masRem[i]];
    else
        sum = sum + masOsn[i];
}
```

Рисунок 2.5 – Вычитание конкретных элементов и последующее полное суммирование.

Описание структур данных

- Библиотеки: <stdio.h>, <stdlib.h>, <math.h>.
- К типу данных “int” относятся такие переменные как “i” и “n”, где “i” – счётчик для цикла “for”, а “n” – количество случайных чисел.
- Переменные “max”, “min”, “sum”, “rem” – относятся к типу данных “double”, где “max” и “min” – максимальный и минимальный возможные случайные числа, “sum” – переменная, отвечающая за значение суммы случайных чисел, а “rem” – приведенная дробная часть.
- Массивы “masOsn” и “masRem” относятся к типу “double” и “int” соответственно. “masOsn” служит для хранения случайных чисел, а “masRem” – для хранения дробной части чисел, приведенной к виду номера элемента.

Описание алгоритмов

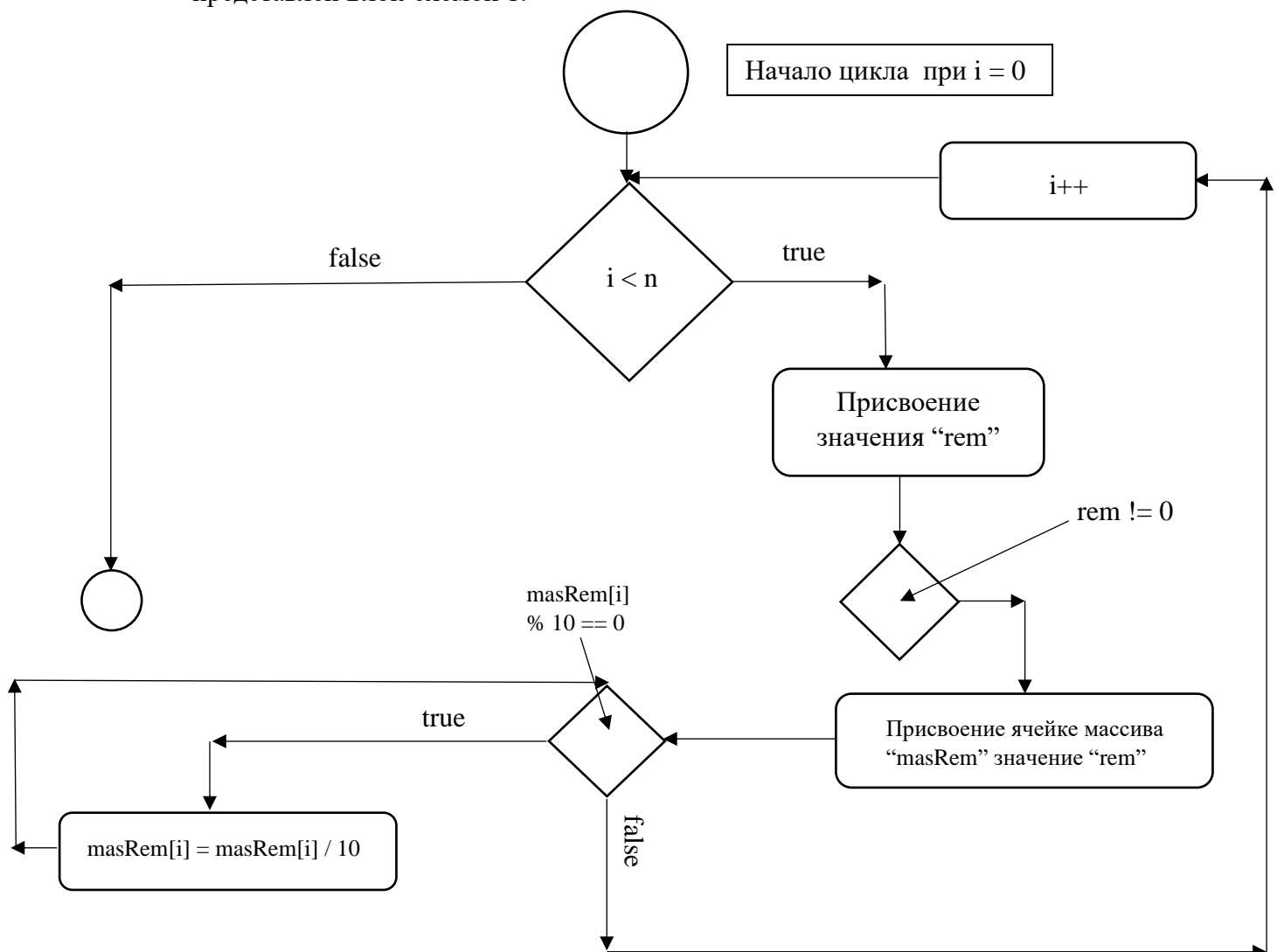
В данной части отчета будут рассмотрены основные алгоритмы, использованные в программе.

1. Объявление массивов “masOsn” и “masRem”, последующее заполнение массива “masOsn” случайными числами, находящимися в промежутке между “max” и “min”. Для этого обратимся к циклу “for”, в котором счетчик “i” будет принимать значения от 0 до “n”, с прибавлением 1 на каждом шаге, и присвоением каждой i’той ячейке массива случайного числа (см. рис 3.1)

```
for (i = 0; i < n; i++)  
{  
    masOsn[i] = (((double)rand() / RAND_MAX) * (max - min) + min);  
}
```

Рисунок 3.1

2. Алгоритм выделения дробной части и последующего преобразования ее в номер представлен Блок-схемой 1.



3. Последующие циклы используются для совершения арифметических действий с некоторой выборкой элементов. С помощью цикла “for”, начинаем проходить по всем элементам массива. Первой операцией является сравнение i ’того номера элемента массива “masRem” на непревышение заданного количества случайных элементов. Далее, при удовлетворении условию “if”, мы вычитаем из переменной “sum” соответствующий элемент, номер которого удовлетворял оператору на предыдущем шаге. В ином исходе (случай “else”) при неудовлетворении условию оператора, мы прибавляем к переменной “sum” данного элемента “masOsn[i]”. (см. рис. 3.2)

```
for (i = 0; i < n; i++)
{
    if (masRem[i] <= n)
        sum = sum - masOsn[masRem[i]];
    else
        sum = sum + masOsn[i];
}
printf("sum = %lf\n", sum);
free(masOsn);
free(masRem);
```

Рисунок 3.2 – Итоговые алгебраические действия.

4. Генерация случайного числа в заданном диапазоне.
- Обратимся опять к рис. 1.1.1 – последняя строчка в котором отвечает за то самое генерирование случайного числа в диапазоне “max”, “min”. Изначально мы задаем случайное число с помощью команды (double)rand(). Следующее действие ((double)rand()/RAND_MAX) отвечает за получение из исходного случайного некоторой дроби, задаваемой, как и исходное число – случайным образом. Следующий шаг ((double)rand()/RAND_MAX * (max - min)) отвечает за приведение дроби, полученной на предыдущем шаге, к случайным значениям в заданном ранее диапазоне (стоит заметить, что на данном этапе мы получаем только положительные значения в диапазоне от min до max, поэтому имеет смысл 4 шаг). Сам же 4 шаг ((double)rand()/RAND_MAX * (max - min) + min) отвечает за то, чтобы сохранялась нижняя граница промежутка (т.е – могли получаться и отрицательные значения при min < 0).

Эксперименты

Любая программа нуждается в тестировании, и эта не является исключением. Протестируем ее в ходе ввода различных значений переменных “n”, “max”, “min”.

- Эксперимент №1

Данный эксперимент нацелен показать работу программы при вводе отрицательного значения переменной “n”(см. прил. 1)

```
Input n, max, min
-1 10 2
Error : check the correctness of the input.
```

Приложение 1. – Ввод отрицательного “n”

- Эксперимент №2

Тоже нацелен на вывод сообщения об ошибке, только теперь при вводе $\text{min} > \text{max}$ (см. прил. 2)

```
Input n, max, min
10 5 10
Error : check the correctness of the input.
```

Приложение 2. – Ввод $\text{max} < \text{min}$.

- Эксперимент №3

Сделаем корректный ввод с небольшими числами (см. прил. 3)

```
Input n, max, min
10 10 5
sum = 70.485855
```

Приложение 3. – Ввод небольших верных значений.

- Эксперимент №4

Введем отрицательные границы для диапазона случайных чисел (см. прил. 4)

```
Input n, max, min
10 -5 -10
sum = -72.762688
```

Приложение 4. – Ввод отрицательных “max” и “min”.

- Эксперимент №5

Введем большие значения для “n”, “max”, “min” (см. прил. 5)

```
Input n, max, min
100 100 20
sum = 3812.996002
```

Приложение 5. – Ввод больших значений для переменных.

- Эксперимент №6

Данный эксперимент выводит в консоли помимо суммы еще и сами случайные числа, а на следующей строке – номер, который взят и преобразован из соответствующей дробной части.

```
Input n, max, min
10 10 5
5.006256
6
7.817927
817
5.966521
966
9.043703
43
7.925047
925
7.399365
399
6.751457
751
9.479812
479
9.114200
114
8.733024
733
sum = 63.734397
```

Приложение 6. – Вывод в консоли случайных чисел и соответствующих номеров.

После проведения данных экспериментов, мы можем убедиться в правильности работы данной программы.

Заключение

В ходе выполнения данной лабораторной работы, с помощью ЯП “С” была написана программа, генерирующая инициализированное пользователем количество случайных чисел в заданном промежутке, после чего, все элементы, номера которых совпадают с дробной частью некоторого элемента вычитаются, а остальные – складываются.

По прошествию выполнения работы мы научились инициализировать динамические массивы типов данных “int” и “double”, работать с циклами “for” и “while”, заполнять массив данными с помощью тех самых циклов, генерировать случайные числа в заданном диапазоне, а также работать с оператором “if”.

Данная лабораторная работа помогла мне лучше разобраться с синтаксисом ЯП “С”, понять структуру генерации случайных чисел, то как можно задавать массивы, и как, собственно, с ними работать.

Литература

- “C Elements of Style” – Режим доступа: [C Elements of Style \(oualline.com\)](http://oualline.com)
- “The C Book” – Эддисон Уэсли. Режим доступа: [The C Book - Table of Contents \(gbdirect.co.uk\)](http://gbdirect.co.uk)
- “Essential C” – Ник Парланте. Режим доступа: [EssentialC.pdf \(stanford.edu\)](http://stanford.edu).

Приложение

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
    int n = 0, i = 0;
    double min = 0.0, max = 0.0, sum = 0.0, rem = 0.0;

    printf("Input n, max, min\n");
    scanf_s("%d %lf %lf", &n, &max, &min);

    if ((n <= 0) || (max <= min)) printf("Error: check the correctness of the
input.");
    else
    {
        double* masOsn = (double*)malloc(sizeof(double) * n);
        int* masRem = (int*)malloc(sizeof(int) * n);

        for (i = 0; i < n; i++)
        {
            masOsn[i] = (((double)rand() / RAND_MAX) * (max - min) + min);
            rem = ((masOsn[i] - (int)masOsn[i]) * 10000);
            if (rem != 0)
            {
                masRem[i] = (int)rem;
                while (masRem[i] % 10 == 0)
                    masRem[i] = masRem[i] / 10;
            }
            //printf("%lf\n", masOsn[i]);
            //printf("%d\n", masRem[i]);
        }
        for (i = 0; i < n; i++)
        {
            if (masRem[i] <= n)
                sum = sum - masOsn[masRem[i]];
            else
                sum = sum + masOsn[i];
        }
        printf("sum = %lf\n", sum);
        free(masOsn);
        free(masRem);
    }
    return 0;
}
```