Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Нижегородский Государственный Университет им.

Н.И.Лобачевского» (ННГУ)

Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работы

Генерация случайных чисел и выполнение операций с ними

Выполнил:

студент группы 3821Б1ПМ3

Сучков В.Н.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород

2021г.

Содержание

[Введение 3](#_Toc85628837)

[Постановка задачи 4](#_Toc85628838)

[Руководство пользователя 5](#_Toc85628839)

[Руководство программиста 6](#_Toc85628840)

[Описание структуры кода программы 6](#_Toc85628841)

[Описание структуры данных 8](#_Toc85628842)

[Описание алгоритмов 8](#_Toc85628843)

[Эксперименты 11](#_Toc85628844)

[Заключение 13](#_Toc85628845)

# Введение

Программирование — это интересный, полезный и увлекательный процесс, благодаря которому создаются программы – набор инструкций, которые приводятся в исполнение компьютерами.

Одной из ключевых задач компьютера является работа с данными. В том числе и со случайно генерируемыми наборами данных, о которых пойдет речь в настоящей работе.

Случайные числа — это одна из основных составляющих любого языка программирования, на них строятся многие алгоритмы. Они имеют применение в физике, например, в исследованиях электронного шума, в инженерном деле и исследовании операций. Многие методы статистического анализа требуют использования случайных чисел.

В ходе выполнения лабораторный работы на языке программирования «С» будет написана программа, работающая со случайными числами.

# Постановка задачи

Программа генерирует множество случайных чисел размера n в диапазоне (min, max), где n, min, max вводятся с клавиатуры. После чего подсчитывает, выводит сумму, которая получается следующим образом: все числа, номера которых совпадают с дробной частью одного из исходных чисел - вычитаются, все остальные прибавляются.

# Руководство пользователя

После запуска программа выводит сообщение «Set N, min, max» означающим, что от пользователя для дальнейшей работы требуется ввести: число элементов массива случайных чисел N, нижнюю границу генерации чисел min и верхнюю max, в одну строку через пробелы, как указано в сообщении от программы.

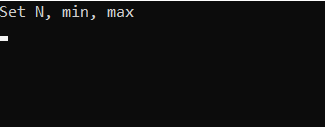


Рисунок 1. Терминал после запуска

Например, введем N = 10, min = 0, max = 99

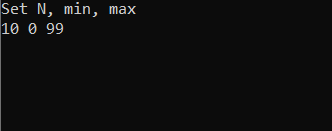


Рисунок 2. Ввод данных от пользователя

После нажатия Enter, программа выведет результат суммирования сгенерированных чисел, произведенного по правилам технического задания. На этом программа завершается.

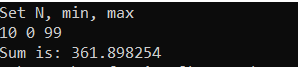


Рисунок 3. Результат работы программы

# Руководство программиста

## Описание структуры кода программы

1)Подключение библиотек, с которыми предстоит работать.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h> |

2)Объявление функции, принимающей на вход число с плавающей запятой и возвращающей первые три знака дробной части в виде целого числа.

|  |
| --- |
| int Fract(float a) //выделение первых 3-х знаков дробной части  {  int b = 0;  b = (int)((a - (int)a)\*1000);  while ((b % 10 == 0) && (b != 0)) {  b = b / 10;  }  if (b >= 0)  return b;  if (b < 0)  return -b;  } |

3)Объявление функции main(), получение данных от пользователя, а также проверка полученных данных на корректность.

|  |
| --- |
| int main()  {  long int N = 0;  float max, min, sum = 0.0;  printf("Set N, min, max\n");  scanf\_s("%d %f %f", &N, &min, &max);  //проверка входных данных на корректность  if (N <= 0)  {  printf("Error: invalid input (N must be more than 0)\n");  return 0;  }  else if ((max - min) <= 0)  {  printf("Error: invalid input (min cannot be equal or more than max)\n");  return 0;  } |

4)Создание динамического массива размера N типа float arrNum и заполнение его случайно сгенерированными в заданном пользователем диапазоне. А также создание динамического массива размера N и типа float arrSum для чисел, которые будут суммироваться в конце программы и массива arrFr, который будет заполнен дробными частями случайных чисел.

|  |
| --- |
| float\* arrNum = (float\*)malloc(sizeof(float) \* N);  float\* arrSum = (float\*)malloc(sizeof(float) \* N);  int\* arrFr = (int\*)malloc(sizeof(int) \* N);  for (int i = 0; i < N; i++)  {  arrNum[i] = rand();  arrNum[i] = (arrNum[i] / RAND\_MAX )\*(max - min) + min;  arrSum[i] = arrNum[i];  arrFr[i] = Fract(arrNum[i]);  } |

5)В случае, если номер какого-либо числа совпадет с дробной частью одного из чисел, то число с таким номером должно вычитаться из итоговой суммы, для этого в первом цикле заменяем такие числа на 0, а в следующим добавляем их с противоположным знаком.

|  |
| --- |
| for (int j = 0; j < N; j++)  {  if (arrFr[j] < N)  {  arrSum[arrFr[j]] = 0;  }  }  for (int j = 0; j < N; j++)  {  if (arrFr[j] < N)  {  arrSum[arrFr[j]] += -arrNum[arrFr[j]];  }  } |

6)Сложение всех элементов arrSum для получения требуемой техническим заданием суммы, вывод результата в терминал, очистка памяти и завершение программы.

|  |
| --- |
| for (int k = 0; k < N; k++)  {  sum = sum + arrSum[k];  }  //вывод суммы и очистка памяти  printf("Sum is: %f", sum);  free(arrNum);  free(arrSum);  free(arrFr);  return 0; |

## Описание структуры данных

1)В программе используются переменные счетчики типа unsigned long int(i, j, k), количество генерируемых чисел N типа long int, а также функция Fract(), возвращающая число типа int.

2)Переменные min, max и sum относятся к типу данных float

3)Случайные числа, создаваемые в ходе работы программы, записываются в массивы типа float arrNum и arrSum, а дробные части этих чисел записываються в целочисленный массив arrFr.

4)А также для функционирования программы требуются библиотеки: stdio.h и stdlib.h.

## Описание алгоритмов

В данной программе реализовано несколько ключевых алгоритмов:

1)Выделение дробной части с фиксированной точностью и представление её как целого числа, реализованный как функция Fract(), принимающая на вход число типа float, и возвращающая дробную часть с точностью три знака после запятой. Для из исходного числа вычитается его целая часть, полученное число умножается на 10 в степени три(количество знаков после запятой), приведение нового числа к типа int и присвоение переменной b значение полученного числа. Затем, чтобы избежать неправильной интерпретации дробной части, делим переменную b на 10, пока это возможно, чтобы убрать лишние нули справа, которые могут получится при умножении на 1000 в предыдущей части алгоритма. Так же, во избежание бесконечного цикла проверяем равно ли b нулю. В конце алгоритма возвращаем значение переменной b если оно неотрицательно и значение переменной -b если оно было отрицательно.

|  |
| --- |
| int Fract(float a) //выделение первых 3-х знаков дробной части  {  int b = 0;  b = (int)((a - (int)a)\*1000);  while ((b % 10 == 0) && (b != 0)) {  b = b / 10;  }  if (b >= 0)  return b;  if (b < 0)  return -b;  } |

2)Алгоритм заполнения массива случайными числами представлен следующей блок-схемой:

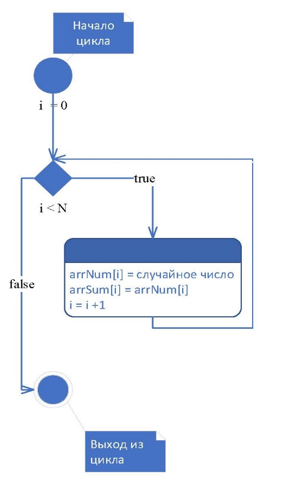


Рисунок 4. Блок-схема заполнения массива случайными числами

3)Алгоритм суммирования. На начальном этапе массив arrSum совпадает с массивом arrNum. В первом цикле все элементы, номера которых совпадут с дробной частью какого-либо числа из arrNum заменяться на 0. В следующем цикле из всех нулевых элементов вычитается соответствующие значение элемента массива arrNum.

Таким образом в итоге массив arrSum содержит: элементы массива arrNum на соответствующих местах, если их номер в массиве arrNum не совпал ни с одной дробной частью какого либо из исходных чисел и противоположные по знаку и умноженные на количество раз, когда их номер совпадает с дробной частью какого либо из исходных чисел элементов массива arrNum. Затем все элементы массива arrSum складываются и полученная сумма выводиться на экран пользователя.

|  |
| --- |
| for (int j = 0; j < N; j++)  {  if (arrFr[j] < N)  {  arrSum[arrFr[j]] = 0;  }  }  for (int j = 0; j < N; j++)  {  if (arrFr[j] < N)  {  arrSum[arrFr[j]] += -arrNum[arrFr[j]];  }  }  for (int k = 0; k < N; k++)  {  sum = sum + arrSum[k];  }  printf("Sum is: %f", sum); |

# Эксперименты

Написанную программу протестируем на различных входных данных, чтобы убедится в её работоспособности в различных условиях.

1)В случае ввода некорректных данных, таких как отрицательное количество элементов или неверный ввод границ диапазона:

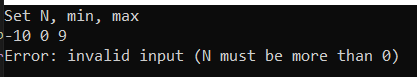


Рисунок 4. Ввод неверного значение N

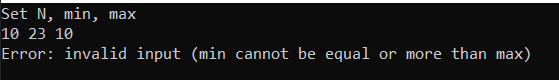


Рисунок 5. Ввод некорректных границ диапазона

2)Работа программы в случае ввода корректных данных:

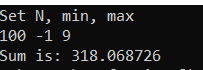


Рисунок 5. Результат работы программы при корректных входных данных

3)Теперь попробуем подать на вход различные корректные значения:

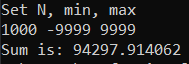


Рисунок 6. Работа программы с отрицательными числами

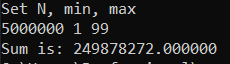


Рисунок 7. Работа программы при больших N



Рисунок 8. Работа программы при больших входных данных

При тестировании программы с различными входными данными в среде разработки Visual Studio никаких ошибок или предупреждений встречено не было. В среднем программа даже для больших входных данных выполнялась за небольшое время, не превышающее четырех секунд для крайне больших значений N.

# Заключение

В ходе лабораторной работы мною была написана программа на языке «С», предназначенная для решения конкретно сформулированной задачи, которая исправно функционирует.

Выполнение данной лабораторной работы позволило мне лучше изучить принципы работы с динамическими массивами, изучить различные алгоритмы позволяющие работать с данными, содержащимися в массивах, принципы работы со случайными числами, принципы работы с вводимыми пользователем данными их реализации внутри программы, их интерпретацию и проверку на корректность.

Выполнение подобных работ позволяет значительно улучшить свои навыки программирования, а также улучшить понимание работы с компьютером и средой разработки, в тоже время позволяет закрепить получаемые знания в области программирования и отработать их применения на практике.