Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Факультет вычислительной математики и кибернетики

Отчёт по лабораторной работе

Упорядочивание массивов и методы поиска данных

Выполнил:

студент ф-та ВМК гр. 81-01

Сиднев А.А.

Проверил:

ассистент каф. МО ЭВМ, ВМК

Сиднев А.А.

Нижний Новгород

2010 г.

Содержание

[Введение 3](#_Toc86201189)

[Постановка задачи 4](#_Toc86201190)

[Руководство пользователя 5](#_Toc86201191)

[Руководство программиста 6](#_Toc86201192)

[Описание структуры программы 6](#_Toc86201193)

[Описание структур данных 7](#_Toc86201194)

[Описание алгоритмов 8](#_Toc86201195)

[Эксперименты 11](#_Toc86201198)

[Заключение 12](#_Toc86201199)

[Литература 13](#_Toc86201200)

[Приложения 14](#_Toc86201201)

[Приложение 1 14](#_Toc86201202)

[Приложение 2 14](#_Toc86201203)

# Введение

Программирование становится неотъемлемой частью жизни многих людей, даже не задумывающихся об этом. Этому увлекательному и интересному занятию

# 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 программы откроется консоль, в которую необходимо ввести указанные данные, а именно:

1. Количество случайных чисел:

|  |
| --- |
|  |

1. Минимальный элемент диапазона:

|  |
| --- |
|  |

1. Минимальный элемент диапазона:

|  |
| --- |
|  |

# Руководство программиста

## Описание структуры программы

1. Подключение необходимых библиотек для работы:

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

1. Объявление функции main(), ввод данных пользователем и их проверка на допустимые значения:

|  |
| --- |
| int main()  {  setlocale(LC\_ALL, "Rus");  int N = 0, i = 0, mant = 0;  float x = 0.0, drob = 0.0, min = 0.0, max = 0.0;  float\* mas;  int\* mantissa;  long double sum = 0.0;  printf("Введите кол-во элементов:\t");  scanf\_s(" \t%d", &N);  if (N <= 0)  {  printf("ERROR");  return 0;  }    printf("Введите минимальный элемент:\t");  scanf\_s(" \t%f", &min);    printf("Введите максимальный элемент:\t");  scanf\_s(" \t%f", &max);  if (min >= max)  {  printf("ERROR");  return 0;  } |

1. Создание двух динамических массивов, размеры которых соответствуют введенному пользователем числу N. В первом будут храниться случайные числа в заданном диапазоне, а во втором будут находится первые 6 цифр после запятой каждого случайного числа из первого массива:

|  |
| --- |
| mas = (float\*)malloc(N \* sizeof(float));  mantissa = (int\*)malloc(N \* sizeof(int)); |

1. Заполнение массивов. Первый содержит случайные вещественные числа в заданном диапазоне. Второй содержит первые шесть цифр после запятой соответствующего числа. Для удобства, представим мантиссу числа в целочисленном виде.

|  |
| --- |
| for (i = 0; i < N; i++)  {  x = (((float)rand()) / RAND\_MAX) \* (max - min) + min;  mas[i] = x;  drob = x - (int)x;  drob = drob \* 1000000;  mant = (int)drob;  if (mant < 0)  mant = -1 \* mant;  mantissa[i] = mant;  } |

1. Проверка на совпадение мантиссы числа с номером любого числа. В случае совпадения - число вычитается из общей суммы, иначе прибавляется:

|  |
| --- |
| for (i = 0; i < N; i++)  {  if ((mantissa[i] < N) && (mantissa[i] >= 0))  sum -= mas[mantissa[i]];  else  sum += mas[i];  } |

1. Вывод полученной суммы, Освобождаем выделенную под массивы память и завершаем программу:

|  |
| --- |
| printf("Полученная сумма:\t%f", sum);  free(mas);  free(mantissa);  return 0;  } |

## Описание структур данных

В программе используются следующие типы данных:

1. «int» - используется для количества элементов (N), счетчика перебора в цикле (i) и мантиссы числа (mant).
2. «float» - используется для вещественного числа (x), дробной части вещественного числа (drob) и границ диапазона (min, max).
3. «float\*» - для создания динамического массива, хранящего в себе вещественные числа.
4. Для хранения случайных чисел используется динамический массив типа «float», для хранения мантиссы чисел используется динамический массив типа «int».
5. «long double» - используется для подсчета суммы.

## Описание алгоритмов

Программа содержит следующие алгоритмы:

1. Заполнение первого массива случайными числами в заданном пользователем диапазоне. Заполнение второго массива дробными частями, переведенными в целочисленный тип «int», чисел из первого массива. Используется 6 цифр после запятой исходного числа. Дли выделения дробной части из случайного числа вычитается его целая часть. Результат умножаем на 1000000, чтобы 6 цифр после запятой, после чего передаем полученное значение в массив предназначенный для мантиссы.

|  |
| --- |
| for (i = 0; i < N; i++)  {  x = (((float)rand()) / RAND\_MAX) \* (max - min) + min;  mas[i] = x;  drob = x - (int)x;  drob = drob \* 1000000;  mant = (int)drob;  if (mant < 0)  mant = -1 \* mant;  mantissa[i] = mant;  } |

# 

# 

i < N

Объявление массивов mas, mantissa

# 

Присваивание случайного числа массиву.

Нахождение мантиссы.

Присваивание мантиссы массиву, содержащему дробные части.

false

true

Рисунок 4. Блок-схема заполнения массива случайными числами

1. Алгоритм нахождения суммы представляет из себя сравнение мантиссы и количества чисел. Если мантисса числа меньше количества всех чисел, тогда номер

# Эксперименты

Результаты экспериментов выполненных вашей программой, а так же их анализ.

# Заключение

Этот пункт содержит перечисление тех результатов, которых вам удалось достигнуть.

# Литература

1. Столлингс, В. Структурная организация и архитектура компьютерных систем, 5-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 896 с.: ил. — Парал. тит. англ.
2. Johnson M. Superscalar Microprocessor Design. — Englewood Cliff, New Jersey: Prentice Hall, 1991.
3. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.: ил.
4. Stone H. High performance Computer Architecture. — Reading, MA: Addison-Wesley, 1993.
5. Tullsen D.M., Eggers S.J. Effective Cache Prefetching on a Bus-Based Multiprocessor. — ACM Transactions on Computer Systems, pp. 57-88, Feb 1995.
6. Chandra D., Guo F., Kim S., Solihin Y. Predicting inter-thread cache contention on a chip multi-processor architecture. — Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA), pp. 340–351, Feb 2005.
7. Press W., Teukolsky S., Vetterling W., Flannery B. Numerical Recipes in C. The Art of Scientific Computing. Second Edition. — Cambridge University Press, 1992.
8. Камаев А.М., Сиднев А.А., Сысоев А.В. Об одном подходе к анализу эффективности приложений // Труды 50-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук»: Часть I. Радиотехника и кибернетика. - М.: МФТИ, 2007.
9. Debugging and performance monitoring. Intel® 64 and IA-32 Architectures Software Developer’s Manual. Volume 3B: System Programming Guide, Part 2. May 2007. — [http://www.intel.com/products/processor/manuals/]
10. Юнаковский А.Д. Начала вычислительных методов для физиков. – Н. Новгород: ИПФ РАН, 2007.

# Приложения

## Приложение 1

## Приложение 2