Федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский Нижегородский государственный университет**

**им. Н.И. Лобачевского»**

**Институт информационных технологий, математики и механики**

Направление подготовки: Прикладная математика и информатика

Отчёт по лабораторной работе

**Виды сортировок массивов и их сравнение**

**Выполнил:** студент группы 3821Б1ПМ3

Благодеров В.А.

**Проверил:** заведующий лабораторией

суперкомпьютерных технологий и

высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород

2021 г.

Содержание

Введение 3

1. Постановка задачи 4

2. Руководство пользователя 5

3. Руководство программиста 6

a. Описание структуры программы 6

b. Описание структуры данных 6

c. Описание алгоритмов 6

4. Эксперименты 10

Заключение 11

Литература 12

Приложения 13

Приложение 1 13

Приложение 2 14

Введение

Программирование занимает одну из самых значительных ниш в современном мире. Это не только способ заставить технику работать, но и поставить себя на путь развития своих способностей. Программирование – процесс создания компьютерных программ, написание инструкций на конкретном языке программирования. Программы способствуют улучшению и облегчения человеческого быта, промышленной деятельности, а также сферы услуг. Для упрощения программирования пользователь использует массивы. Сортировка массива – процесс принятия и расположение отдельных элементов массива в определённом виде логической последовательности в соответствии с рядом правил, которые определены пользователем. Процесс включает в себя шаги через массив и тестирование элемента против окружающих элементов, чтобы определить, требуется ли его переместить в другой индекс в массиве. При выполнении сортировки массива, существует несколько алгоритмов, которые могут использоваться. Большинство массивов, имеют алгоритмы сортировки, которые измеряют их скорость и эффективность, с помощью самых медленных алгоритмов и самого простого для программирования, поэтому самый быстрый гораздо сложнее.

1. Постановка задачи

Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных случайно определяется преподавателем) задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла.

Реализовать сортировки: пузырьком, вставкой, быстрая.

Сравнить время работы, сделать выводы.

Первая программа создаёт текстовый файл с записанными в него числами. Программа принимает количество чисел N, максимальное и минимальное значение.

Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

1. Руководство пользователя

Чтобы воспользоваться первой программой(Lab2), необходимо ввести три числа N, min, max, где N-размер массива, min – минимальное число и max – максимальное число (Рис.1).

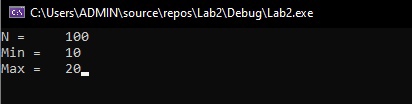


Рис.1 Ввод начальных значений

При попытке ввести некорректные данные программа выдаст ошибку (Рис.2-3).

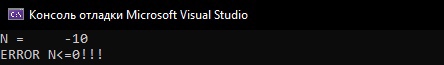


Рис.2 Неправильный ввод значения N

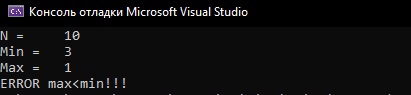


Рис.3 Неправильный ввод значений min и max

Чтобы воспользоваться второй программой(Lab2.2), необходимо вводить значения соответствующие консольному интерфейсу (Рис. 4).

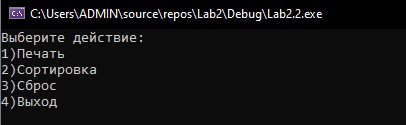


Рис.4 Консольный интерфейс

1. Руководство программиста
   1. Описание структуры программы

Программа состоит из двух проектов Lab2 и Lab2.2. Используется главная функция main(). При выполнении консольного приложения, написанного на языке Си, операционная система компьютера передаёт управление функции с именем main(). Функцию main() нельзя вызывать из других функций программы, она является управляющей. Другие используемые функции: PrintMas() – печать массива, bubbleSort() – сортировка пузырьком, insertionSort() – сортировка вставкой, quickSort() – быстрая сортировка, qSort() – вызов функции быстрой сортировки.

* 1. Описание структуры данных

Массив типа данных float.

* 1. Описание алгоритмов

Программа реализует сортировки: пузырьком, вставкой, быстрая.

1. (Пузырёк) Суть метода. Все соседние элементы массива попарно сравниваются друг с другом и меняются местами в том случае, если предшествующий элемент больше последующего. В результате этого максимальный элемент постепенно смещается вправо и занимает крайнее правое место в массиве, после чего не обрабатывается. Затем процесс повторяется до тех пор, пока вся последовательность не будет упорядочена (Смотри блок схему 1).

temp=mas[j-1]

mas[j-1]=mas[j]

mas[j]=temp

j=N-1,i

mas[j-1] >mas[j]

Нет

Да

i=0, N

Блок-схема 1. «Сортировка пузырёк»

1. (Вставка) Суть метода. Просматриваются элементы массива, начиная со второго. Каждый новый элемент вставляется на подходящее место в уже упорядоченную последовательность. Это место определяется последовательными сравнениями элемента с упорядоченными элементами (Смотри блок схему 2).

i=1,N

mas[j]<temp

temp=mas[i]

j=i-1

mas[j+1]=mas[j]

mas[j]=temp

Нет

Да

Блок-схема 2. «Сортировка вставкой»

1. (Быстрая) Суть метода. В исходном неотсортированном массиве выбрать некоторый элемент x (барьерный элемент). Переставить элементы массива таким образом, чтобы слева от x оказались элементы, меньшие или равные x, а справа – элементы, большие x, разделив таким образом массив, можно сделать тоже самое с обеими полученными частями, а затем – с частями этих частей и т.д., пока каждая часть будет содержать только один элемент (Смотри блок схему 3).

Блок-схема 3. «Быстрая сортировка»

while (mas[i] < x)

i++;

while (mas[j] > x)

j--;

if (i <= j)

{

temp = mas[i];

mas[i] = mas[j];

mas[j] = temp;

i++;

j--;

}

i<=j

f<j

i<L

Да

Нет

quickSort(mas, i, l);

quickSort(mas, f, j);

Да

Да

Нет

1. Эксперименты

Проведём несколько экспериментов. Возьмём разное количество элементов с диапазоном от 1 до 10. Результаты занесем в таблицу.

Таблица 1. «Сравнение сортировок»

|  |  |  |
| --- | --- | --- |
| **Вид сортировки** | **Количество элементов, N** | **Время сортировки, сек** |
| Пузырёк | 1000 | 0,003 |
| Пузырёк | 10000 | 0,403 |
| Пузырёк | 100000 | 39,829 |
| Вставка | 1000 | 0,001 |
| Вставка | 10000 | 0,165 |
| Вставка | 100000 | 16,601 |
| Быстрая | 1000 | 0,000 |
| Быстрая | 10000 | 0,002 |
| Быстрая | 100000 | 0,016 |

Заключение

В ходе лабораторной работы мною была выполнена поставленная задача. Мне удалось реализовать сортировки: пузырьком, вставкой, быстрая. Сделав эксперименты можно сделать вывод о том, что пузырьковая сортировка и сортировка вставками подходят для сортировки небольшого набора данных. Оба метода имеют более низкую эффективность по сравнению с быстрой сортировкой.

Литература

1. Брайан Керниган, Деннис Ритчи. Язык программирования Си. — Москва: Финансы и статистика, 1992. — 272 с.

Приложения

Приложение 1

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <malloc.h>  int main()  {  int N = 0, min = 0, max = 0, i = 0;  float\* mas = 0;  printf("N =\t");  scanf\_s("%d", &N);  if (N <= 0)  {  printf("ERROR N<=0!!!");  return 0;  }  printf("Min =\t");  scanf\_s("%d", &min);  printf("Max =\t");  scanf\_s("%d", &max);  if (max < min)  {  printf("ERROR max<min!!!");  return 0;  }  mas = (float\*)malloc(N \* sizeof(float));  if (mas == 0)  {  printf("ERROR mas1 == 0!!!");  return 0;  }  for (i = 0; i < N; i++)  {  mas[i] = ((float)rand() / RAND\_MAX) \* (max - min) + min;  //printf("mas[%d] = %f\n", i, mas[i]);  }  FILE\* file = 0;  fopen\_s(&file, "Masmain.txt", "w");  fprintf(file, "N = %d\n", N);  for (i = 0; i < N; i++)  {  fprintf(file, "%f\n", mas[i]);  }  fclose(file);  free(mas);  return 0;  } |

Приложение 2

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <malloc.h>  #include <locale.h>  #include <time.h>  void PrintMas(float\* mas, int N)  {  int i = 0;  for (i = 0; i < N; i++)  {  printf("mas[%d] = %f\n", i, mas[i]);  }  }  void bubbleSort(float\* mas, int N)  {  int i, j;  float temp = 0;  for (i = 0; i < N; i++)  {  for (j = N - 1; j >= i; j--)  {  if (mas[j - 1] > mas[j])  {  temp = mas[j - 1];  mas[j - 1] = mas[j];  mas[j] = temp;  }  }  }  }  void insertionSort(float\* mas, int N)  {  int i, j;  float temp;  for (i = 1; i < N; i++)  {  temp = mas[i];  for (j = i - 1; j >= 0; j--)  {  if (mas[j] < temp)  break;  mas[j + 1] = mas[j];  mas[j] = temp;  }  }  }  void quickSort(float\* mas, int f, int l)  {  int i = f;  int j = l;  float temp = 0;  float x = mas[(int)((f + l) / 2)];  do  {  while (mas[i] < x)  i++;  while (mas[j] > x)  j--;  if (i <= j)  {  temp = mas[i];  mas[i] = mas[j];  mas[j] = temp;  i++;  j--;  }  } while (i <= j);  if (i < l)  quickSort(mas, i, l);  if (f < j)  quickSort(mas, f, j);  }  void qSort(float\* mas, int N)  {  quickSort(mas, 0, N - 1);  }  int main()  {  int N = 0, i = 0, a = 0;  float\* mas = 0;  float\* mas2 = 0;  FILE\* file = 0;  fopen\_s(&file, "C:\\Users\\ADMIN\\source\\repos\\Lab2\\Lab2\\Masmain.txt", "r");  fscanf\_s(file, "N = %d\n", &N);  mas = (float\*)malloc(N \* sizeof(float));  mas2 = (float\*)malloc(N \* sizeof(float));  for (i = 0; i < N; i++)  {  fscanf\_s(file, "%f\n", &(mas[i]));  }  fclose(file);  for (i = 0; i < N; i++)  mas2[i] = mas[i];  setlocale(LC\_ALL, "Russian");  printf("Выберите действие:\n1)Печать\n2)Сортировка\n3)Сброс\n4)Выход\n");  while (a != 4)  {  scanf\_s("%d", &a);  if (a == 1)  {  PrintMas(mas, N);  }  else  if (a == 2)  {  printf("Выбери сортировку:\n1)Пузырьком\n2)Вставкой\n3)Быстрая\n");  scanf\_s("%d", &a);  if (a == 1)  {  clock\_t t1 = clock();  bubbleSort(mas, N);  clock\_t t2 = clock();  PrintMas(mas, N);  printf("Время работы = %lf\n", ((double)(t2 - t1))/ CLOCKS\_PER\_SEC);  }  else  if (a == 2)  {  clock\_t t1 = clock();  insertionSort(mas, N);  clock\_t t2 = clock();  PrintMas(mas, N);  printf("Время работы = %lf\n", ((double)(t2 - t1)) / CLOCKS\_PER\_SEC);  }  else  if (a == 3)  {  clock\_t t1 = clock();  qSort(mas, N);  clock\_t t2 = clock();  PrintMas(mas, N);  printf("Время работы = %lf\n", ((double)(t2 - t1))/ CLOCKS\_PER\_SEC);  }  else  printf("Такой цифры тут нет\n");  }  else  if (a == 3)  {  for (i = 0; i < N; i++)  mas[i] = mas2[i];  printf("Сброс выполнен\n");  }  if (a == 4)  printf("Выход успешно выполнен");  }  free(mas);  free(mas2);  return 0;  } |