Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н. И. Лобачевского

Институт информационных технологий математики и механики

Отчёт по лабораторной работе

Генерирование случайных чисел. Сортировка случайных чисел.

Выполнил:

студент 1-го курса ИИТММ гр. 3821Б1ПМ3

Мерзляков В.А

Проверил:

Заведующий лабораторией

суперкомпьютерных технологий и

высокопроизводительных

вычислений

Лебедев И. Г.

Нижний Новгород

2021 г.

Содержание

1.[Введение 3](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962758)

2.[Постановка задачи 4](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962759)

3.[Руководство пользователя 5](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962760)-6

4.[Руководство программиста 7](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962761)

4.1.[Описание структуры программы 7](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962762)

4.2.[Описание алгоритмов 7-10](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962763)

5.[Эксперименты 11](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962765)

6.[Заключение 12](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962766)

7.[Литература 13](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.[Приложение 14](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.1.[Приложение 1 14](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

8.2.[Приложение 2 15-17](file:///C:\Users\User\Downloads\Шаблон%20отчёта%20по%20Лабораторной%20работе.doc#_Toc270962767)

2

# 1. Введение

На сегодняшний день программирование играет ведущую роль в жизни.

Цель программирования – создание оптимизированных алгоритмов для решения различного рода задач, не обязательно связанных с компьютерами. Программы создаются с помощью специальных языков программирования, имеющих маленькое сходство с естественными языками. Они созданы для того, чтобы интегрировать команды человека в машинный код, то есть «от человека к компьютеру».

Язык «C» является фундаментом для любого программиста, так как все остальные языки берут своё начало именно с него. Он является одним из самых популярных языков для программирования.

В языке «C» важную играет использование массивов. Для представления набора однотипных данных используются массивы, являющиеся наряду с циклами фундаментальным инструментом программирования. Массивы позволяют избегать дублирования кода и разрабатывать компактные, понятные и красивые программы.

Для наглядного примера и более детального разбора принципа работы массивов обратимся к моей лабораторной работе, написанной на языке «С».

Программа написана на английском языке, так как при попытке русифицировать текст программы работали некорректно.

3

**2. Постановка задачи**

Задача лабораторной работы такова: написать две программы, в первой из которых идет создание текстового файла с записанными в него случайными числами. Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход) и выполняет выбранные пользователем действия.

Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “double”) задаваемых: обязательно случайно, дополнительно с клавиатуры.

Реализовать сортировки: пузырьком, вставкой, быстрая.

Сравнить время работы, сделать выводы.

4

**3. Руководство пользователя.**

Для лабораторной работы используется две программы. При запуске первой программы, она запросит ввести количество чисел. Затем выбор стоит за пользователем, как ввести эти числа: вручную или использовать генерацию случайных чисел.

После запуска первой программы на экран выведется сообщение: «Введите количество чисел».(Рис.1)

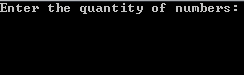


Рис.1 Вывод окна консоли.

После ввода количества чисел(например 10) программа попросит вас выбрать способ ввода чисел: «Нажмите 1, если хотите ввести числа вручную» и «Нажмите 2, если хотите получить случайные числа.(Рис.2)

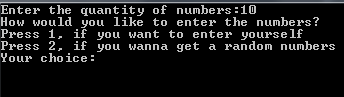


Рис.2 Выбор пользователя для получения случайных чисел.

После выбора способа (например 2) программа остановится.(Рис.3)

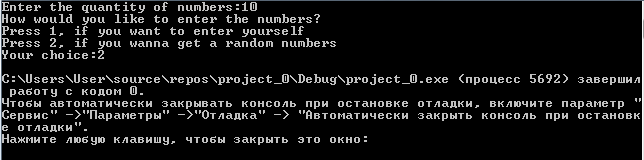
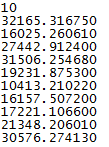


Рис.3 Окончание работы программы.

После окончания работы программы, нужно открыть файл, и, если все правильно, то в файле появятся числа. Первое число – количество чисел, все остальные – результат работы программы.(Рис.4)



5

Рис.4 Полученный результат в файле.

После завершения работы первой программы следует запустить вторую. После её запуска на экран будет выведен консольный интерфейс с выбором команд.(Рис.5)

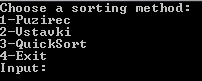


Рис.5 Вывод консоли

Консоль выводит на экран следующие команды:

1. «Puzirec»
2. «Vstavki»
3. «QuickSort»
4. «Exit»

Команда 1 отвечает за сортировку пузырьком.

Команда 2 отвечает за сортировку вставкой.

Команда 3 отвечает за быструю сортировку.

Команда 4 отвечает за преждевременное окончание работы программы.

После выбора команды программа выведет на экран отсортированные числа соответствующим способом, время сортировки.(Рис.6)

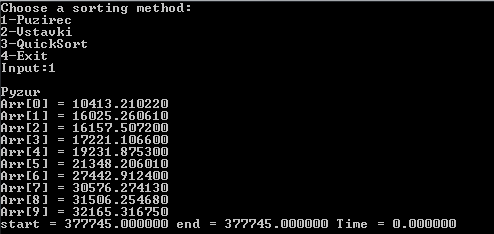
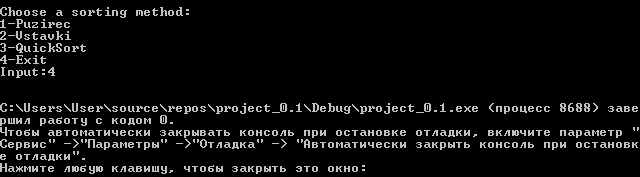


Рис.6 Результат работы программы.

Аналогичным образом выглядит экран при вызове команд 2 и 3.

При вызове команды 4 программа завершит свою работу.(Рис.7)



6

Рис.7 Завершение работы программы.

**4. Руководство программиста**

**4.1 Описание структуры программы.**

Первая программа состоит из одного модуля int main() {…}, в котором находится код генерирования случайных чисел и записи их в файл.

Вторая программа также состоит из одного модуля int main() {…}, в котором находится код сортировки чисел различными способами.

Обе программы написаны без использования вспомогательных файлов для кодировки, поэтому все функции и их вызовы находятся непосредственно в программах.

Использованные функции:

Puzirec – использует метод сортировки «пузырьком»

Vstavki – использует метод сортировки «вставка»

qSort – использует «быструю сортировку»

**4.2 Описание алгоритмов.**

1)Алгоритм создание случайных чисел и записи их в файл. У пользователя есть выбор как получить случайные числа – вручную или с помощью генерации случайных чисел.(Рис.8)

|  |
| --- |
| kek = fopen("C:/Users/User/source/repos/project\_0.1/project\_0.1/input.txt","w");  printf\_s("Enter the quantity of numbers:");  scanf\_s("%d", &n);  fprintf(kek, "%d\n", n);  printf\_s("How would you like to enter the numbers?\nPress 1, if you want to enter yourself\nPress 2, if you wanna get a random numbers\nYour choice:");  scanf\_s("%d", &k);  if (k == 1)  {  double a = 0.0;  for (int i = 0; i < n; i++)  {  scanf\_s("%lf", &a);  fprintf(kek, "%lf\n", a);  }  }  else  {  for (int i = 0; i < n; i++)  {  double a = rand();  int b = rand();  while (b>1)  {  c += b % 10;  b = b / 10;  c = c \* 0.1;  }  a = a + c;  fprintf(kek, "%lf\n", a);  c = 0;  } |

Рис.8 Фрагмент кода(1) Создание случайных чисел, их запись в файл 7

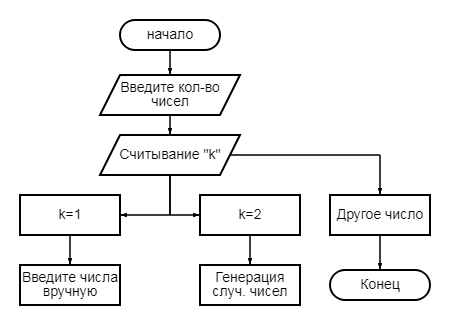
****

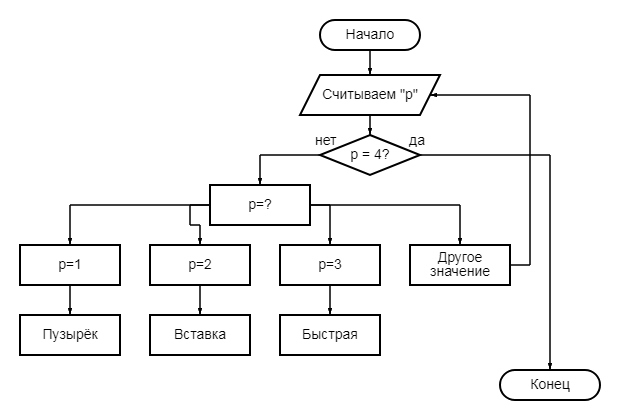
Рис.9 Блок-схема(1) консоли первой программы

**2) Алгоритм считывания чисел из файла**(Рис.10)

|  |
| --- |
| fin = fopen("input.txt", "r");  fscanf\_s(fin, "%d", &n);  double\* Arr = NULL;  Arr = (double\*)malloc(n \* sizeof(double));  for (int i = 0; i < n; i++)  {  fscanf\_s(fin, "%lf", &Arr[i]);  } |

Рис.10 Фрагмент кода(2) Считывание чисел из файла.

**3) Алгоритм в консоли второй программы**(Рис.11)



8

Рис.11 Блок-схема(2).

**4) Алгоритм сортировки пузырьком**(Рис.12)

|  |
| --- |
| void Puzirec(double\* num, int size, clock\_t\* s, clock\_t\* e)  {  \*s = clock();  for (int i = 0; i < size - 1; i++)  {  for (int j = (size - 1); j > i; j--)  {  if (num[j - 1] > num[j])  {  double temp = num[j - 1];  num[j - 1] = num[j];  num[j] = temp;  }  }  }  \*e = clock();  } |

Рис.12 Фрагмент кода(3) Сортировка пузырьком

**5) Алгоритм сортировки вставкой**(Рис.13)

|  |
| --- |
| void Vstavki(double\* num, int size, clock\_t\* s, clock\_t\* e)  {  double newElement;  int location;  \*s = clock();  for (int i = 1; i < size; i++)  {  newElement = num[i];  location = i - 1;  while (location >= 0 && num[location] > newElement)  {  num[location + 1] = num[location];  location = location - 1;  }  num[location + 1] = newElement;  }  \*e = clock();  } |

Рис.13 Фрагмент кода(4) Сортировка Вставкой

9

**6) Алгоритм быстрой сортировки**(Рис.14)

|  |
| --- |
| void qSort(double\* data, int first, int last)  {  double temp = 0;  if (first < last)  {  int left = first, right = last;  double middle = data[(left + right) / 2];  do  {  while (data[left] < middle)  {  left++;  }  while (data[right] > middle)  {  right--;  }  if (left <= right)  {  temp = data[left];  data[left] = data[right];  data[right] = temp;  left++;  right--;  }  } while (left < right);  qSort(data, first, right);  qSort(data, left, last);  }  } |

Рис.14 Фрагмент кода(5) Быстрая сортировка

10

**5. Эксперименты.**

|  |  |  |
| --- | --- | --- |
| **Способ сортировки** | **Количество чисел** | **Время сортировки, мс** |
| Пузырёк | 10000 | ≈198 |
| Вставка | 10000 | ≈149 |
| Быстрая | 10000 | ≈4 |
| Пузырёк | 100000 | ≈34497 |
| Вставка | 100000 | ≈2408 |
| Быстрая | 100000 | ≈31 |

Исходя из результатов, можно сделать вывод, что быстрее всего работает «быстрая сортировка». На втором месте находится сортировка «вставка». Ну и самым неоптимизированным методом сортировки является «пузырёк».

11

**6. Заключение**

В ходе выполнения лабораторной работы мною были написаны 2 программы на языке программирования «С», которые полностью выполняют поставленные задачи, а именно:

1) Создание определенного количества случайных чисел

2) Запись их в файл

3) Считывание чисел из файла

4) Реализация сортировки чисел(тип данных «double») пузырьком, вставкой и быстрой сортировкой.

5) Сравнение скорости работы сортировок, установление времени их работы.

Первая программа генерирует случайные числа и записывает их в файл. Способ ввода чисел стоит за пользователем – ввести вручную или сгенерировать случайные числа.

Вторая программа читает текстовый файл с набором чисел и выводит консоль, где пользователь может выбрать способ сортировки или преждевременно завершить работу программы.

Интерфейс программы простой, однако все сообщения, выводящиеся на экран написаны на английском языке. Однако люди даже с самым маленьким уровнем владения смогут её понять.

Исходя из результатов лабораторной работы можно сделать вывод, что «Quick Sort» является самым быстрым способом сортировки с любым типом данных, так как даже с огромным количеством чисел он справляется за доли секунды. Сортировка «Вставка» значительно медленнее, однако аутсайдером является сортировка «Пузырьком», которой можно сортировать только небольшие количества чисел.

12

**7. Список литературы**

1. Т.А. Павловская Учебник по программированию на языках высокого  
уровня(С/С++) – Режим доступа: <http://cph.phys.spbu.ru/documents/First/books/7.pdf>  
2. Бьерн Страуструп. Язык программирования С++ - Режим доступа:  
<http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf>

13

**8. Приложения**

***8.1 Приложение 1***

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <locale.h>  #include <time.h>  #include<string.h>  int main()  {  int n = 0;  int k = 0;  double c = 0;  FILE\* kek;  srand(time(NULL));  kek = fopen("C:/Users/User/source/repos/project\_0.1/project\_0.1/input.txt", "w");  printf\_s("Enter the quantity of numbers:");  scanf\_s("%d", &n);  fprintf(kek, "%d\n", n);  printf\_s("How would you like to enter the numbers?\nPress 1, if you want to enter yourself\nPress 2, if you wanna get a random numbers\nYour choice:");  scanf\_s("%d", &k);  if (k == 1)  {  double a = 0.0;  for (int i = 0; i < n; i++)  {  scanf\_s("%lf", &a);  fprintf(kek, "%lf\n", a);  }  }  else  {  for (int i = 0; i < n; i++)  {  double a = rand();  int b = rand();  while (b>1)  {  c += b % 10;  b = b / 10;  c = c \* 0.1;  }  a = a + c;  fprintf(kek, "%lf\n", a);  c = 0;  }  }  return 0;  } |

14

***8.2 Приложение 2***

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <locale.h>

#include <time.h>

void Puzirec(double\* num, int size, clock\_t\* s, clock\_t\* e)

{

\*s = clock();

for (int i = 0; i < size - 1; i++)

{

for (int j = (size - 1); j > i; j--)

{

if (num[j - 1] > num[j])

{

double temp = num[j - 1];

num[j - 1] = num[j];

num[j] = temp;

}

}

}

\*e = clock();

}

void Vstavki(double\* num, int size, clock\_t\* s, clock\_t\* e)

{

double newElement;

int location;

\*s = clock();

for (int i = 1; i < size; i++)

{

newElement = num[i];

location = i - 1;

while (location >= 0 && num[location] > newElement)

{

num[location + 1] = num[location];

location = location - 1;

}

num[location + 1] = newElement;

}

\*e = clock();

}

void qSort(double\* data, int first, int last)

{

double temp = 0;

if (first < last)

{

int left = first, right = last;

double middle = data[(left + right) / 2];

do

{

while (data[left] < middle)

{

left++;

}

while (data[right] > middle)

{

right--;

}

if (left <= right) 15

{

temp = data[left];

data[left] = data[right];

data[right] = temp;

left++;

right--;

}

} while (left < right);

qSort(data, first, right);

qSort(data, left, last);

}

}

int main()

{

int p = 0;

int n = 0;

clock\_t start, end;

while (p != 4)

{

FILE\* fin;

fin = fopen("input.txt", "r");

fscanf\_s(fin, "%d", &n);

double\* Arr = NULL;

Arr = (double\*)malloc(n \* sizeof(double));

for (int i = 0; i < n; i++)

{

fscanf\_s(fin, "%lf", &Arr[i]);

}

printf("Choose a sorting method:\n1-Puzirec\n2-Vstavki\n3-QuickSort\n4-Exit\nInput:");

scanf("%d", &p);

printf("\n");

if (p == 1)

{

Puzirec(Arr, n, &start, &end);

printf("Pyzur\n");

for (int i = 0; i < n; i++)

printf("Arr[%d] = %lf\n", i, Arr[i]);

printf("start = %lf end = %lf Time = %lf\n", (double)start, (double)end, (double)(end - start) / CLOCKS\_PER\_SEC);

printf("-------\n");

}

if (p == 2)

{

printf("Vstavka\n");

Vstavki(Arr, n, &start, &end);

for (int i = 0; i < n; i++)

printf("Arr[%d] = %lf\n", i, Arr[i]);

printf("start = %lf end = %lf Time = %lf\n", (double)start, (double)end, (double)(end - start) / CLOCKS\_PER\_SEC);

printf("-------\n");

}

if (p == 3)

{

printf("QuickSort\n");

start = clock();

qSort(Arr, 0, n-1);

end = clock();

for (int i = 0; i < n; i++)

printf("Arr[%d] = %lf\n", i, Arr[i]);

16

printf("start = %lf end = %lf Time = %lf\n", (double)start, (double)end, (double)(end - start) / CLOCKS\_PER\_SEC);

printf("-------\n");

}

if (p == 4) break;

}

return 0;

}

17