Федеральное государственное образовательное бюджетное учреждение  
высшего профессионального образования  
«Нижегородский Государственный Университет им.  
Н.И.Лобачевского» (ННГУ)

Национальный исследовательский Университет  
Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе  
Сортировки.

Выполнил:  
студент группы 3821Б1ФИ3  
Козырева.Е.А

Проверил:

Нижний Новгород  
2021 г.

**Содержание.**

[Введение…………………………………………………………………………………………. 3](#_Введение.)

[Постановка задачи………………………………………………………………………………. 4](#_Toc86237500)

[Руководство пользователя……………………………………………………………………… 5](#_Toc86237501)

[Руководство программиста……………………………………………………………………... 8](#_Toc86237502)

[Эксперименты………………………………………………………………………………….. 14](#_Toc86237503)

[Заключение……………………………………………………………………………………... 19](#_Toc86237504)

[Литература……………………………………………………………………………………… 20](#_Toc86237505)

[Приложение…………………………………………………………………………………….. 21](#_Toc86237506)

# Введение.

Цель: реализовать программу для сортировки массива.

Задача: уметь использовать разные сортировки для массива любого объёма.

Главное, что надо знать:

1. Виды сортировок (пузырьком, вставкой, быстрая).
2. Таблица символов ASCII.
3. Случайные числа — это последовательность чисел, которая подчиняется одному из законов распределения.
4. Генератор псевдослучайных чисел (ГПСЧ, англ. pseudorandom number generator, PRNG) — алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному).

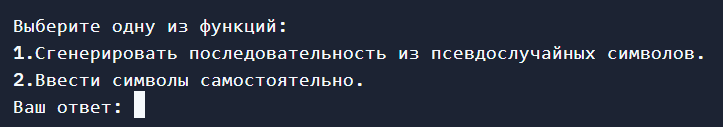
# Постановка задачи.

1. Сравнение сортировок (пузырьком, вставкой, быстрая).
2. Реализовать сортировки массивов данных (тип “char”), задаваемых пользователем (ввод с клавиатуры, файл, случайная генерация).
3. Сравнить время работы сортировок, сделать выводы.
4. Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение.
5. Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

# Руководство пользователя.

**Первая программа.**

Для начала необходимо выбрать тип ввода данных.

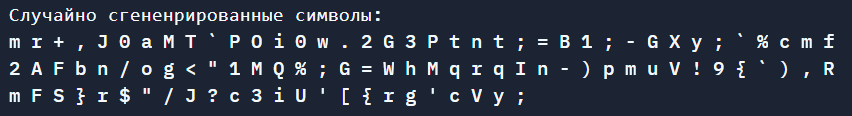


* При ответе 1.

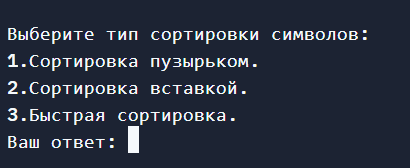
1. После необходимо ввести количество символов для генерации.



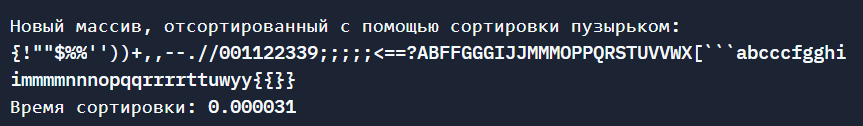
1. Программа выводит сгенерированные символы.



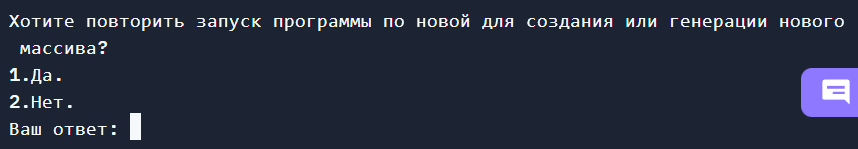
1. Выбираем тип сортировки.



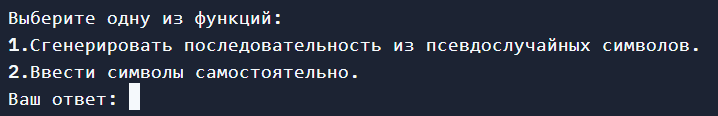
* При выборе сортировки пузырьком.



После чего программа спрашивает о запуске программы по новой.



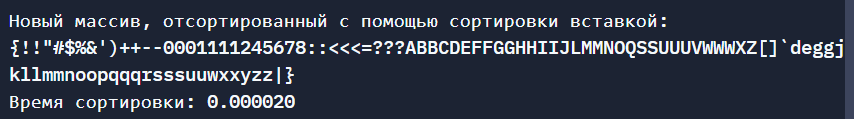
* При ответе 1.



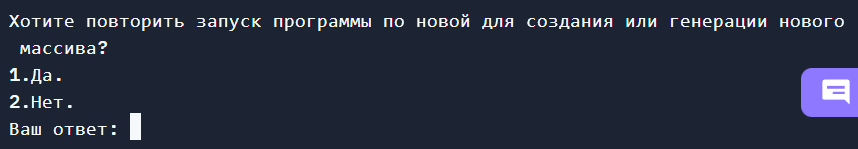
* При ответе 2.



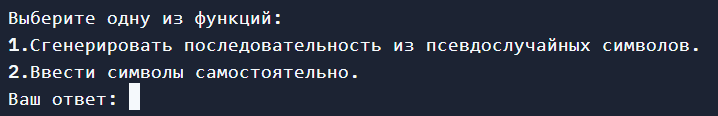
* При выборе сортировки вставкой.



После чего программа спрашивает о запуске программы по новой.



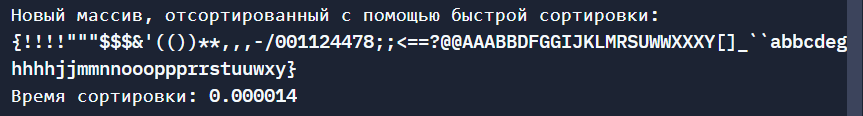
* При ответе 1.



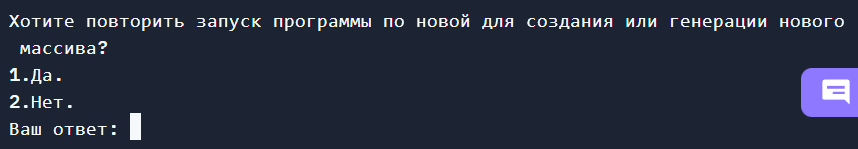
* При ответе 2.



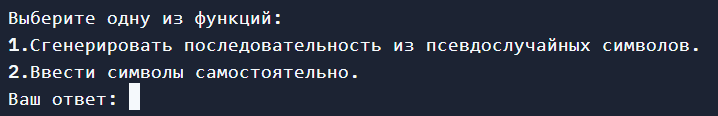
* При выборе быстрой сортировки.



После чего программа спрашивает о запуске программы по новой.



* При ответе 1.



* При ответе 2.

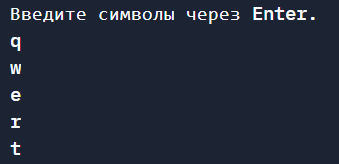


* При ответе 2.

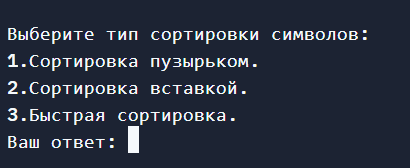
1. После необходимо ввести количество символов для генерации.



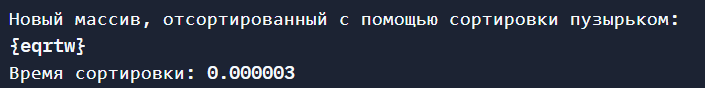
1. Вводим символы с клавиатуры.



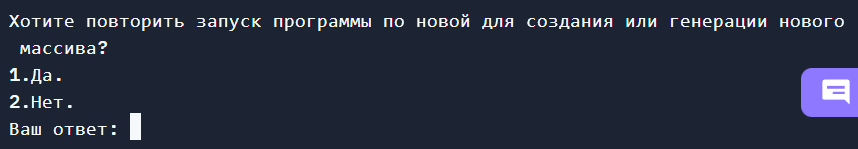
1. Выбираем тип сортировки.



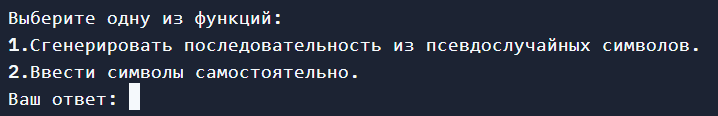
* При выборе сортировки пузырьком.



После чего программа спрашивает о запуске программы по новой.



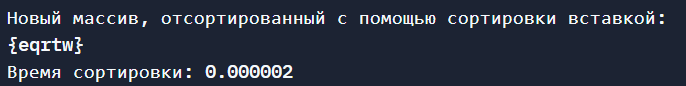
* При ответе 1.



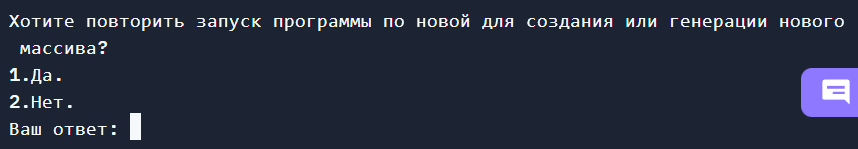
* При ответе 2.



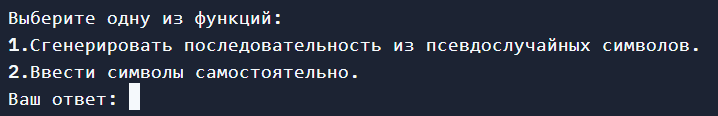
* При выборе сортировки вставкой.



После чего программа спрашивает о запуске программы по новой.



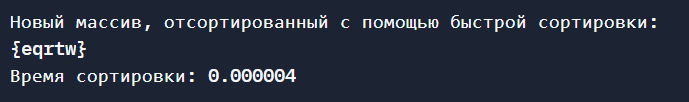
* При ответе 1.



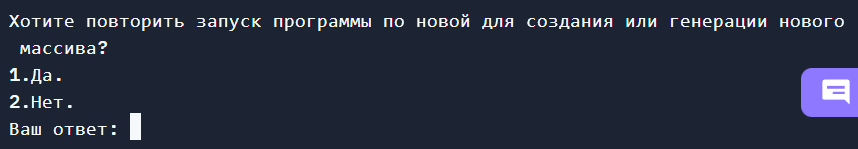
* При ответе 2.



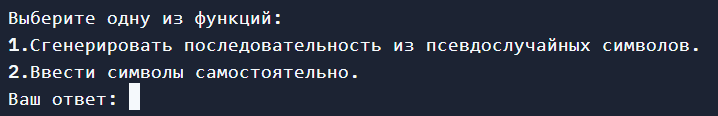
* При выборе быстрой сортировки.



После чего программа спрашивает о запуске программы по новой.



* При ответе 1.

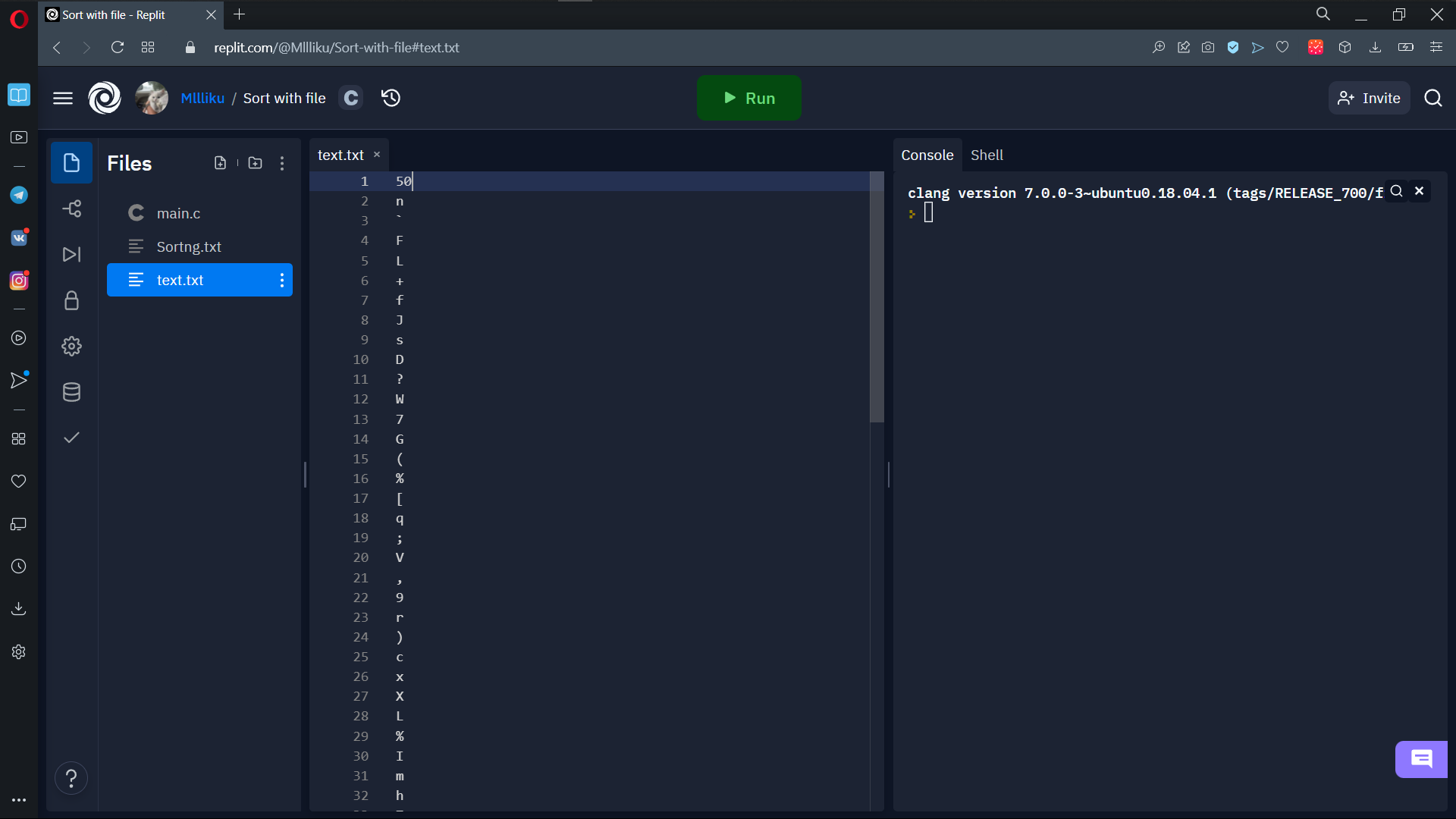


* При ответе 2.



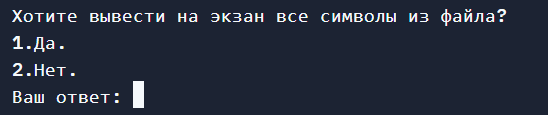
**Вторая программа.**

Вводим данные в файл “text.txt”.

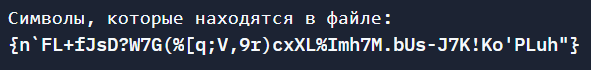


Переходим в файл “main.c” и запускаем программу.

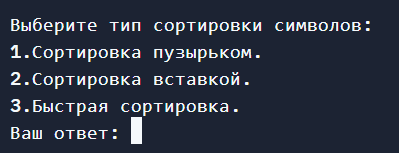
Для проверки правильного считывания символов из файла и их правильного ввода, программа спрашивает о выводе символов на экран.



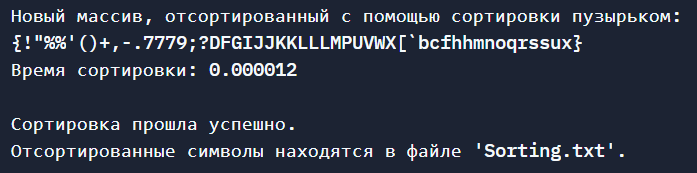
* При ответе 1.



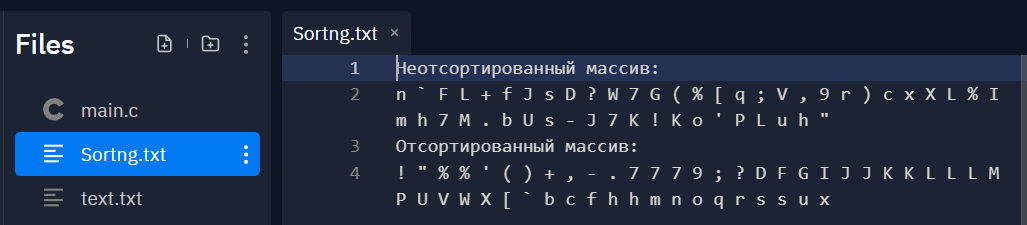
После выбираем тип сортировки.



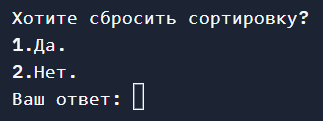
1. При выборе сортировки пузырьком.



Для легкости копирования отсортированные символы записываются в файл “Sorting.txt”.



Сброс сортировки или нет.

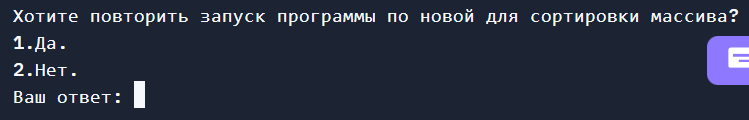


* При ответе 1.

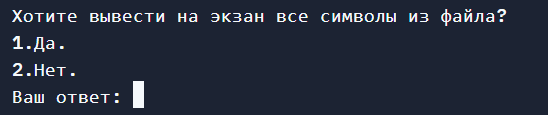


После чего программа запускается по новой.

* При ответе 2.



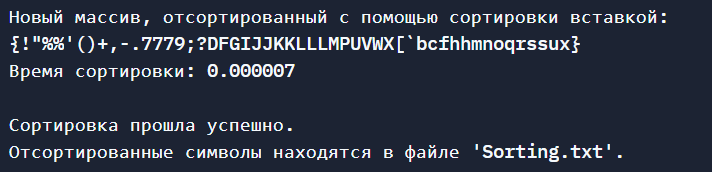
* При ответе 1, программа запустит программу по новой, где надо будет выбрать новую сортировку.



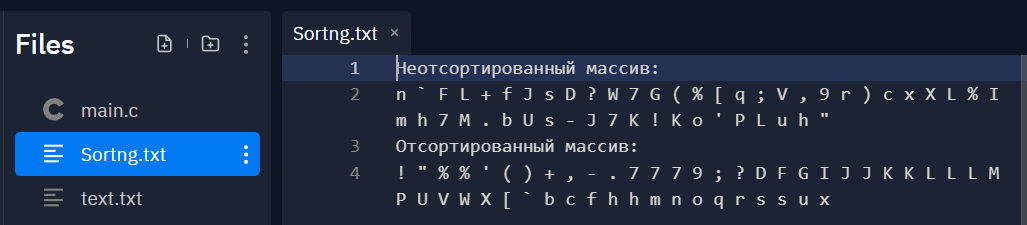
* При ответе 2, программа завершает свою работу.



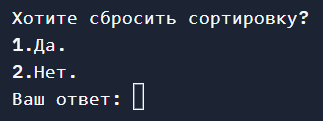
1. При выборе сортировки вставкой.



Для легкости копирования отсортированные символы записываются в файл “Sorting.txt”.



Сброс сортировки или нет.

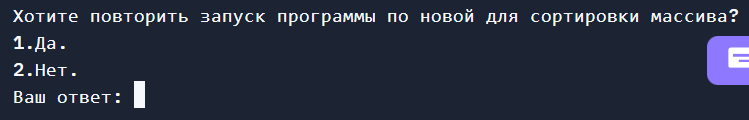


* При ответе 1.

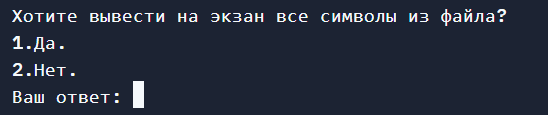


После чего программа запускается по новой.

* При ответе 2.



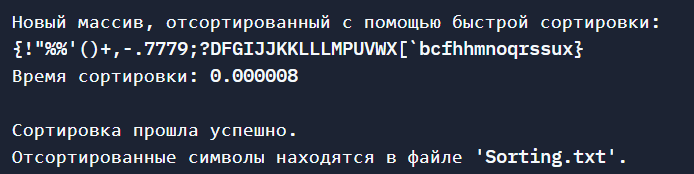
* При ответе 1, программа запустит программу по новой, где надо будет выбрать новую сортировку.



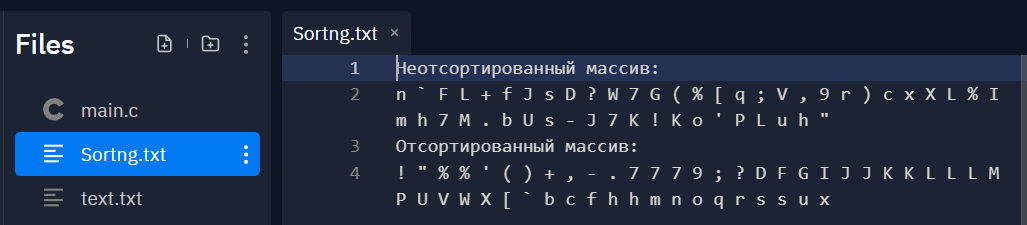
* При ответе 2, программа завершает свою работу.



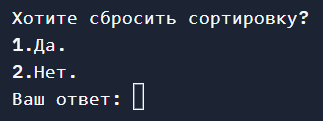
1. При выборе быстрой сортировки.



Для легкости копирования отсортированные символы записываются в файл “Sorting.txt”.



Сброс сортировки или нет.

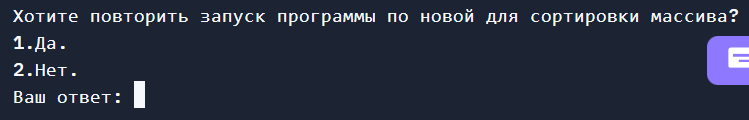


* При ответе 1.

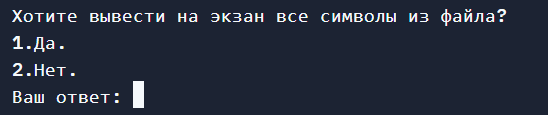


После чего программа запускается по новой.

* При ответе 2.



* При ответе 1, программа запустит программу по новой, где надо будет выбрать новую сортировку.

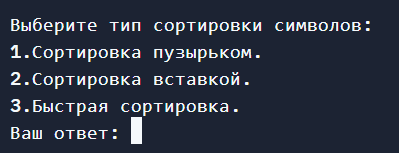


* При ответе 2, программа завершает свою работу.

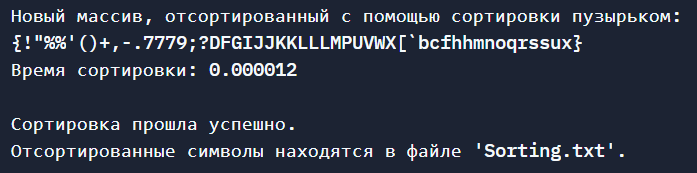


* При ответе 2.

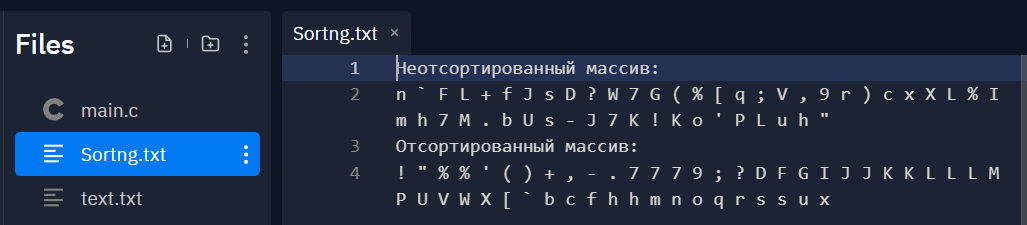
После выбираем тип сортировки.



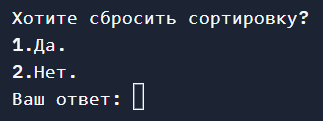
1. При выборе сортировки пузырьком.



Для легкости копирования отсортированные символы записываются в файл “Sorting.txt”.



Сброс сортировки или нет.

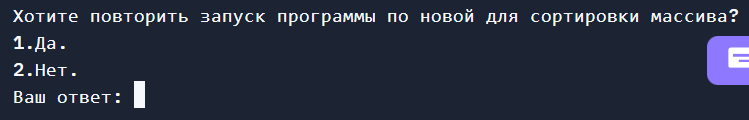


* При ответе 1.

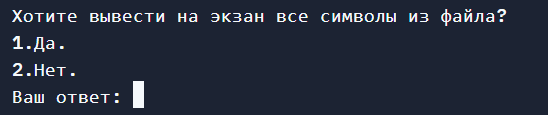


После чего программа запускается по новой.

* При ответе 2.



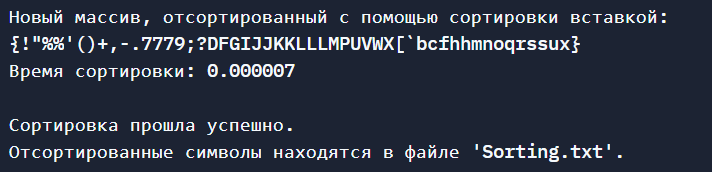
* При ответе 1, программа запустит программу по новой, где надо будет выбрать новую сортировку.



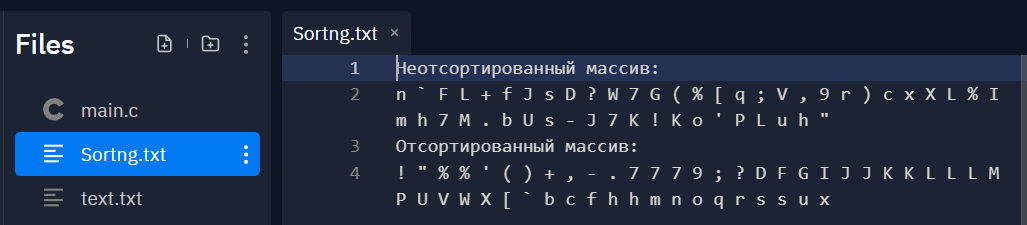
* При ответе 2, программа завершает свою работу.



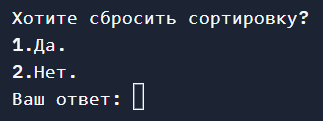
1. При выборе сортировки вставкой.



Для легкости копирования отсортированные символы записываются в файл “Sorting.txt”.



Сброс сортировки или нет.

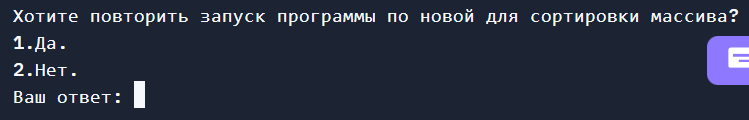


* При ответе 1.

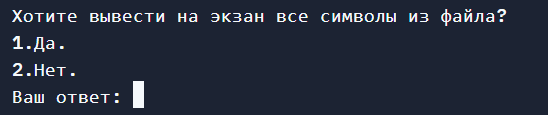


После чего программа запускается по новой.

* При ответе 2.



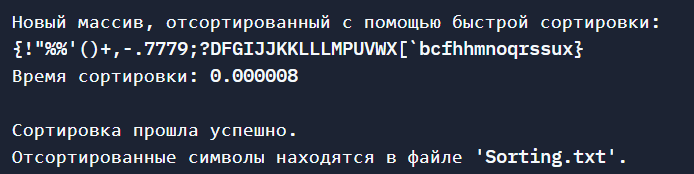
* При ответе 1, программа запустит программу по новой, где надо будет выбрать новую сортировку.



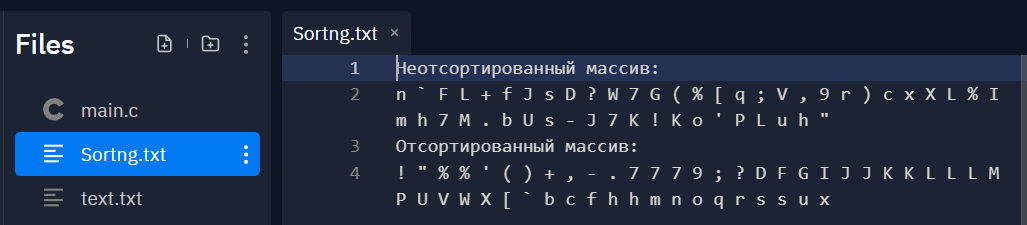
* При ответе 2, программа завершает свою работу.



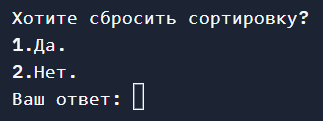
1. При выборе быстрой сортировки.



Для легкости копирования отсортированные символы записываются в файл “Sorting.txt”.



Сброс сортировки или нет.

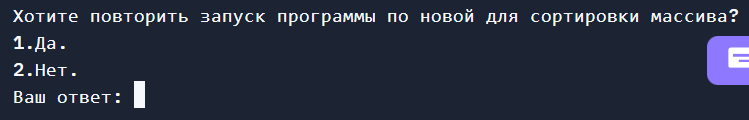


* При ответе 1.

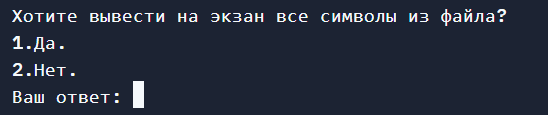


После чего программа запускается по новой.

* При ответе 2.



* При ответе 1, программа запустит программу по новой, где надо будет выбрать новую сортировку.



* При ответе 2, программа завершает свою работу.



# Руководство программиста.

***Описание структуры программы.***

**Первая программа.**

Библиотеки:

* #include <stdio.h> - заголовочный файл стандартной библиотеки языка Си, содержащий определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода.
* #include <string.h> - заголовочный файл стандартной библиотеки языка Си, содержащий функции для работы со строками, оканчивающимися на 0, и различными функциями работы с памятью.
* #include <stdlib.h> - заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контролем процесса выполнения программы, преобразованием типов и другие.
* #include <stdbool.h> - заголовочный файл стандартной библиотеки языка программирования С, который содержит четыре макроса для работы с типом данных bool.
* #include <time.h> - заголовочный файл стандартной библиотеки языка программирования C, содержащий типы и функции для работы с датой и временем.

Переменные:

Тип int.

* answer – для считывания ответа пользователя.
* element/random\_element – для считывания элемента с файла/генерации.
* index/location/circle – индексация для массивов и циклов.
* count\_elements/count – кол-во элементов в массиве.
* flag\_for\_file – для переноса данных в файл.
* number – для проверки символа по таблице ASCII.
* Begin/end and left/right – для индексации в быстрой сортировки.
* tmp – для сохранения символа во время сортировки.

Тип bool.

* flag – для сортировки пузырьком.

Массивы:

* array – основной массив во всех сортировках.
* array\_of\_random\_characters/array\_of\_characters – массивы с полученными символами.
* sorted\_random\_array/sorted\_array – отсортированные массивы по таблице ASCII.

Функции:

* type\_of\_sort() – выбор типа сортировки.
* transferring\_data\_to\_file() – передача данных в файл.
* bubble\_sort() – сортировка пузырьком.
* sorting\_by\_insertion() – сортировка вставкой.
* quick\_sort() – быстрая сортировка.
* launching\_the\_program() – запуск программы по новой.
* main() – база программы.

Файлы:

* Sorting.txt – массив до сортировки и после сортировки.

**Вторая программа.**

Библиотеки:

* #include <stdio.h> - заголовочный файл стандартной библиотеки языка Си, содержащий определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода.
* #include <string.h> - заголовочный файл стандартной библиотеки языка Си, содержащий функции для работы со строками, оканчивающимися на 0, и различными функциями работы с памятью.
* #include <stdlib.h> - заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контролем процесса выполнения программы, преобразованием типов и другие.
* #include <stdbool.h> - заголовочный файл стандартной библиотеки языка программирования С, который содержит четыре макроса для работы с типом данных bool.
* #include <time.h> - заголовочный файл стандартной библиотеки языка программирования C, содержащий типы и функции для работы с датой и временем.

Переменные:

Тип int.

* answer – для считывания ответа пользователя.
* element– для считывания элемента с файла.
* index/location/circle – индексация для массивов и циклов.
* count\_elements/count – кол-во элементов в массиве.
* flag\_for\_file – для переноса данных в файл.
* number – для проверки символа по таблице ASCII.
* Begin/end and left/right – для индексации в быстрой сортировки.
* tmp – для сохранения символа во время сортировки.

Тип bool.

* flag – для сортировки пузырьком.

Массивы:

* array – основной массив во всех сортировках.
* array\_of\_characters – массив с полученными символами.
* sorted\_array – отсортированный массив по таблице ASCII.

Функции:

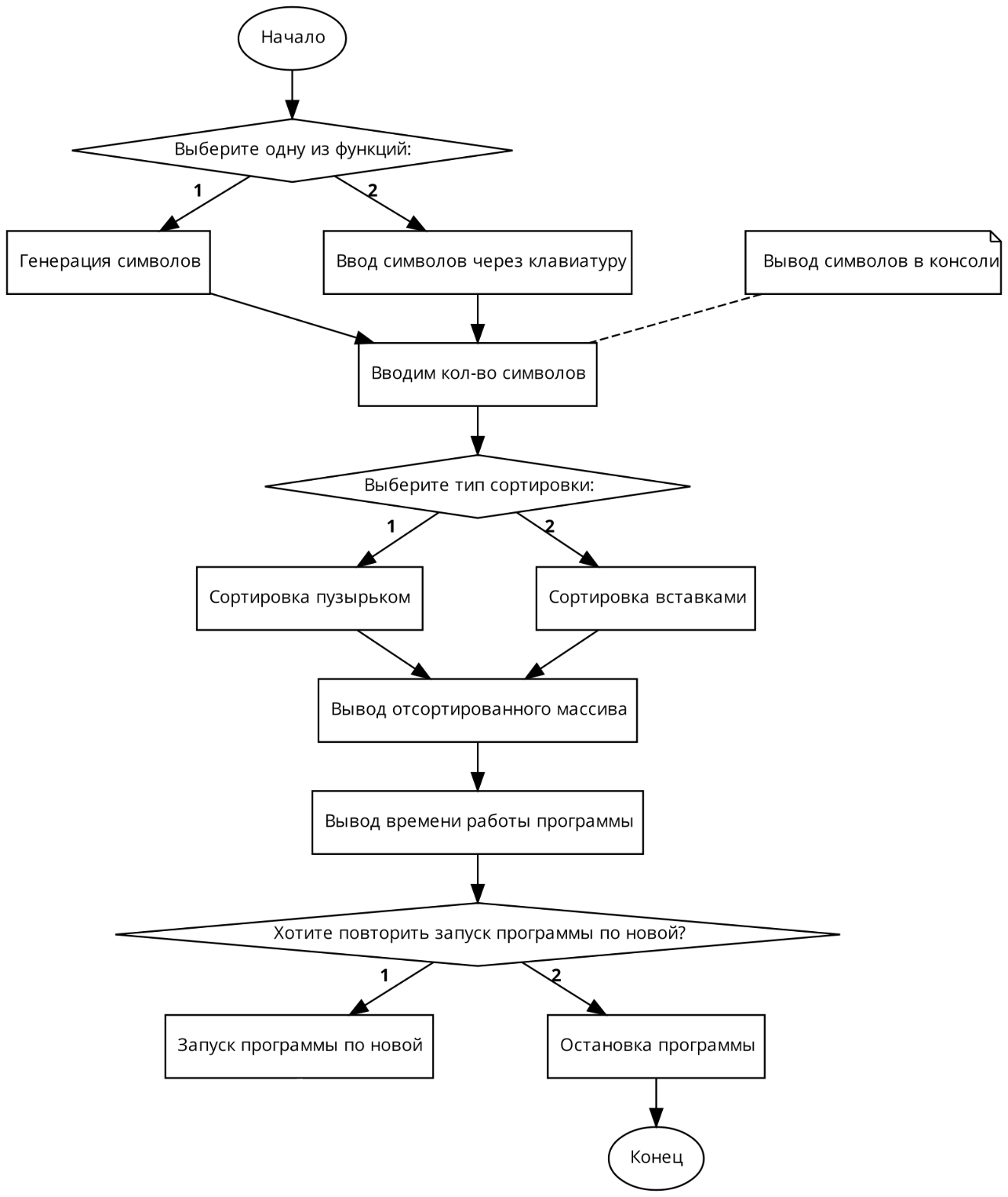
* type\_of\_sort() – выбор типа сортировки.
* output\_data\_from\_file() – выгрузка данных из файла.
* transferring\_data\_to\_file() – передача данных в файл.
* bubble\_sort() – сортировка пузырьком.
* sorting\_by\_insertion() – сортировка вставкой.
* quick\_sort() – быстрая сортировка.
* reset\_sorting() – сброс сортировки массива.
* launching\_the\_program() – запуск программы по новой.
* main() – база программы.

Файлы:

* Sorting.txt – массив до сортировки и после сортировки.
* Text.txt – файл с основными данными.

***Описание алгоритмов.***

**Первая программа.**

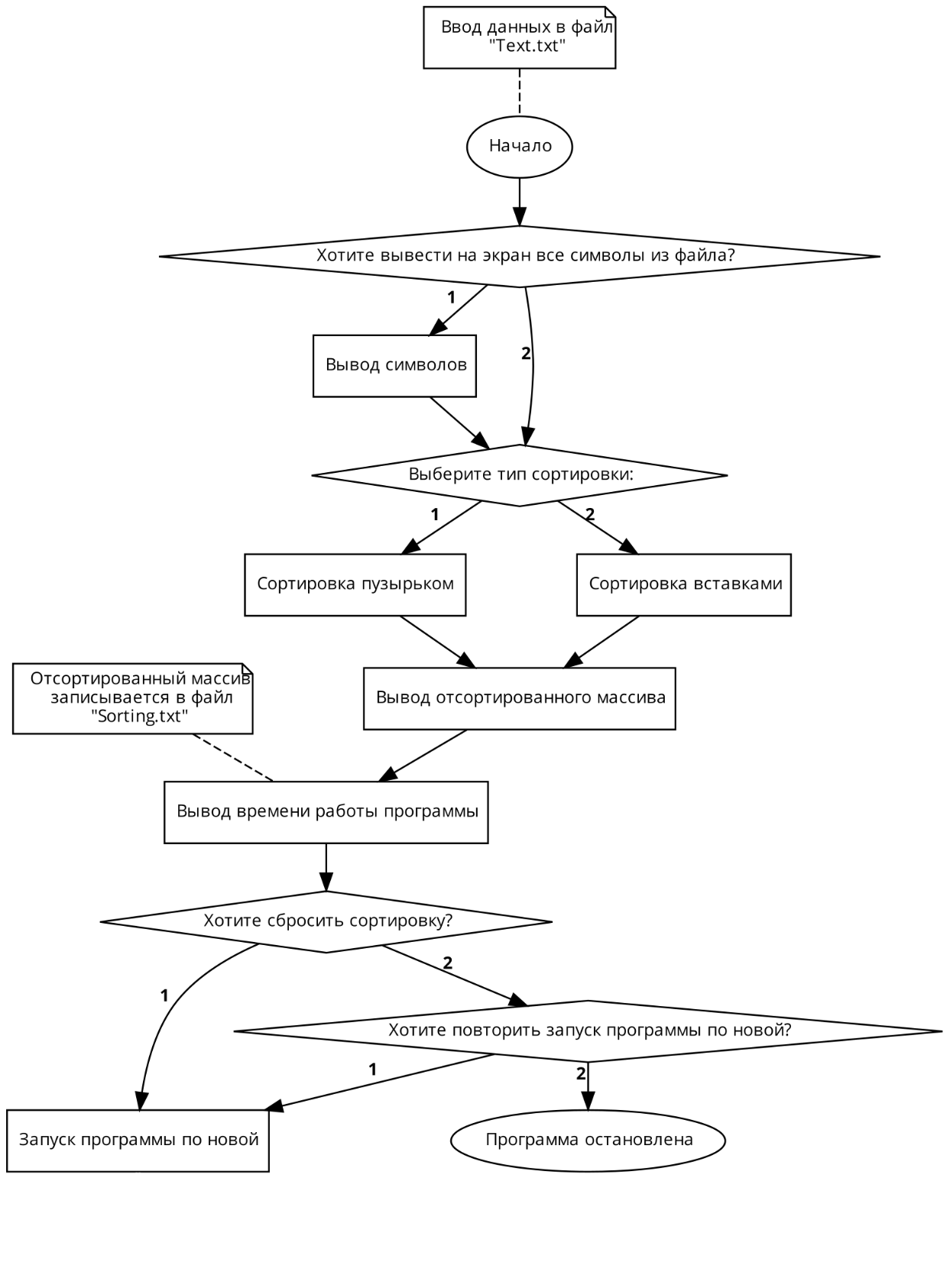
******

Запись отсортированного массива в файл

**3**

Быстрая сортировка

**Вторая программа.**



Быстрая сортировка

**3**

# Эксперименты.

**Ввод с клавиатуры.**

|  |  |  |
| --- | --- | --- |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 5 | 0.000003 |
| Сортировка вставкой | 5 | 0.000002 |
| Быстрая сортировка | 5 | 0.000003 |
|  | | |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 10 | 0.000003 |
| Сортировка вставкой | 10 | 0.000003 |
| Быстрая сортировка | 10 | 0.000004 |

**Генерация символов.**

|  |  |  |
| --- | --- | --- |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 10 | 0.000005 |
| Сортировка вставкой | 10 | 0.000003 |
| Быстрая сортировка | 10 | 0.000003 |
|  | | |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 100 | 0.000056 |
| Сортировка вставкой | 100 | 0.000033 |
| Быстрая сортировка | 100 | 0.000014 |
|  | | |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 1000 | 0.002907 |
| Сортировка вставкой | 1000 | 0.001358 |
| Быстрая сортировка | 1000 | 0.000121 |
|  | | |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 10000 | 0.329345 |
| Сортировка вставкой | 10000 | 0.140037 |
| Быстрая сортировка | 10000 | 0.001173 |

**Данные из файла.**

|  |  |  |
| --- | --- | --- |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 5 | 0.000003 |
| Сортировка вставкой | 5 | 0.000002 |
| Быстрая сортировка | 5 | 0.000003 |
|  | | |
| Сортировка | Кол-во элементов | Время сортировки |
| Сортировка пузырьком | 10 | 0.000003 |
| Сортировка вставкой | 10 | 0.000003 |
| Быстрая сортировка | 10 | 0.000004 |

# Заключение.

**Сортировка пузырьком.**

Плюсы:

* Не требуется дополнительных массивов.
* Очень лёгкий в реализации.
* Минимальное количество ошибок.

Минусы:

* Алгоритм считается учебным и почти не применяется на практике из-за низкой эффективности.
* Медленно работает на тестах, в которых маленькие элементы стоят в конце массива.

**Сортировка вставками.**

Плюсы:

* Алгоритм эффективен при работе со массивами.
* Алгоритм отлично справляется с массивами небольшого размера.
* Может работать с последовательно поступающими данными.

Минусы:

* Очень много перемещений элементов массива.
* Высокая алгоритмическая сложность N².

**Быстрая сортировка.**

Плюсы:

* Является одним из самых эффективных и часто используемых методов.

Минусы:

* Может оказаться малоэффективной на массивах, состоящих из небольшого числа элементов.
* Алгоритм неустойчив.
* Использование рекурсии в неверно составленном коде может привести к переполнению стека.

Литература.

1. Ввод данных из файла и вывод в файл - <https://younglinux.info/c/fopen>
2. Работа с текстовыми файлами - <https://learnc.info/c/text_files.html>
3. Работа с файлами - <https://prog-cpp.ru/c-files/>
4. Чтение и запись текстовых файлов - <https://metanit.com/cpp/c/7.4.php#:~:text=%7D%20Вначале%20открываем%20файл%20на,и%20произойдет%20выход%20из%20цикла>
5. Лекции по C/C++: работа с файлами (fstream) - <http://blog.kislenko.net/show.php?id=1402>
6. #9 - Работа с файлами через язык Си - <https://itproger.com/course/c-programming/9>
7. Потоковый ввод-вывод в файлы - <https://prog-cpp.ru/cpp-files/>
8. Функция strncat - <http://cppstudio.com/post/698/>
9. Создание и запись в файл - <https://www.cyberforum.ru/c-beginners/thread1095405.html>
10. Работа с текстовыми файлами - <http://fsweb.info/programming/c/textfiles.html>
11. Работа с файлами в С - <https://3ty.ru/languages/language-c/rabota-s-faylami-v-c>
12. Форматированный вывод - <https://learnc.info/c/formatted_input_output.html>
13. Локализация и кириллица в консоли - <https://metanit.com/cpp/c/1.5.php>
14. Функция для вычисления длины массива в Си - <https://ru.stackoverflow.com/questions/1202263/Функция-для-вычисления-длины-массива-в-Си>
15. Язык Си в примерах/ASCII-коды символов - <https://ru.wikibooks.org/wiki/Язык_Си_в_примерах/ASCII-коды_символов>
16. Язык Си в примерах/Сортировка - <https://ru.wikibooks.org/wiki/Язык_Си_в_примерах/Сортировка>
17. Таблица символов ASCII - <https://snipp.ru/handbk/table-ascii>
18. Руководство по языку программирования C - <https://metanit.com/cpp/c/>
19. C Standard Library Reference Tutorial - <https://www.tutorialspoint.com/c_standard_library/index.htm>
20. Генерация случайных чисел в языке Си - <https://youngcoder.ru/lessons/4/sluchainie_chisla_na_c.php>
21. Строки в языке C - <https://server.179.ru/tasks/cpp/total/051.html>
22. Обработка строк - <https://learnc.info/c/mastering_strings.html>

# Приложение.

1. **Первая программа.**

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <stdbool.h>  #include <time.h>  // Выбираем сортировку  int type\_of\_sort()  {  int answer;  printf("\nВыберите тип сортировки символов:\n");  printf("1.Сортировка пузырьком.\n");  printf("2.Сортировка вставкой.\n");  printf("3.Быстрая сортировка.\n");  printf("Ваш ответ: ");  scanf("%d", &answer);  if (answer == 1 || answer == 2 || answer == 3)  {  return answer;  }  else  {  printf("Неверный ввод, введите 1 или 2 или 3.\n");  return type\_of\_sort();  }  }  // Передаём данные с массивов в файл  int transferring\_data\_to\_file(int flag, int count, int array[])  {  FILE \*file;  if (flag == 0)  {  file = fopen("Sortng.txt", "w");  fprintf(file, "Неотсортированный массив:\n");  for( int index = 0; index < count; index++)  {  for (int number = 33; number <= 126; number++)  if (array[index] == number)  {  fprintf(file, "%c ", array[index]);  }  }  fprintf(file, "\n");  fclose(file);  }  else if (flag == 1)  {  file = fopen("Sortng.txt", "a+");  fprintf(file, "Отсортированный массив:\n");  for( int index = 0; index < count ; index++)  {  for (int number = 33; number <= 126; number++)  {  if (array[index] == number)  {  fprintf(file, "%c ", array[index]);  }  }  }  fclose(file);  }  return 0;  }  // Сортировка пузырьком  int bubble\_sort(int count, int array[])  {  int element;  bool flag;  clock\_t time\_start= clock();  for (int circle = count - 1; circle >= 0; circle--)  {  flag = 1;  for (int index = 0; index < circle; index++)  {  if (array[index] > array[index + 1])  {  element = array[index];  array[index] = array[index + 1];  array[index + 1] = element;  flag = 0;  }  }  if (flag == 1)  {  break;  }  }  clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, (count \* 2), array);    // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью сортировки пузырьком:\n{");  for (int index = 0; index < (count \* 2); index++)  {  for (int number = 33; number <= 126; number++)  if ((int)array[index] == number)  {  printf("%c", array[index]);  }  }  printf("}\n");    printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);  return 0;  }  // Сортировка вставкой  int sorting\_by\_insertion(int count, int array[])  {  int element, location;  clock\_t time\_start= clock();    for (int index = 1; index < count; index++)  {  element = array[index];  location = index - 1;  while(location >= 0 && array[location] > element)  {  array[location+1] = array[location];  location = location - 1;  }  array[location+1] = element;  }  clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, count, array);  // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью сортировки вставкой:\n{");  for (int index = 0; index < count; index++)  {  for (int number = 33; number <= 126; number++)  if ((int)array[index] == number)  {  printf("%c", array[index]);  }  }  printf("}\n");    printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);  return 0;  }  // Быстрая сортирровка  int quick\_sort(int array[], int begin, int end)  {  if (begin < end)  {  int left = begin, right = end, middle = array[(left + right) / 2];  do  {  while (array[left] < middle)  {  left++;  }  while (array[right] > middle)  {  right--;  }  if (left <= right)  {  int tmp = array[left];  array[left] = array[right];  array[right] = tmp;  left++;  right--;  }  }  while (left <= right);  {  quick\_sort(array, begin, right);  quick\_sort(array, left, end);  }  }  return 0;  }  // Запуск программы по новой  int launching\_the\_program()  {  int answer;  printf("\nХотите повторить запуск программы по новой для создания или генерации нового массива?\n");  printf("1.Да.\n2.Нет.\nВаш ответ: ");  scanf("%d", &answer);  if (answer == 1 || answer == 2)  {  return answer;  }  else  {  printf("Неверный ввод, введите 1 или 2.\n");  return launching\_the\_program();  }  }  // Основная программа  int main()  {  // Ответ пользователя при выборе функции  int answer;  /\* Кол-во символов при генерации псевдослучайных символов  или при считывании символов с клавиатуры \*/  int count\_elements;  // Возможные функции программы  printf("\nВыберите одну из функций:\n");  printf("1.Сгенерировать последовательность из псевдослучайных символов.\n");  printf("2.Ввести символы самостоятельно.\n");  printf("Ваш ответ: ");  scanf("%d", &answer);  switch (answer)  {  case 1:  {  srand(time(NULL));    // Основные данные для функционала программы  printf("\nВведите кол-во символов, которые хотите получить: ");  scanf("%d", &count\_elements);  // Массив, в который будут передаваться данные пользователя  int array\_of\_random\_characters[count\_elements \* 2];  int sorted\_random\_array[count\_elements \* 2];  // Переменная для сгенерированного символа  char random\_element;  printf("Случайно сгененрированные символы:\n");  for (int index = 0; index < count\_elements; index++)  {  random\_element = 33 + rand()%93;  printf("%c ", random\_element);  array\_of\_random\_characters[index] = random\_element;  }  printf("\n");  // Убираем из массива ненужные символы  for (int index = 0; index < count\_elements; index++)  {  int element;  element = (int)array\_of\_random\_characters[index];  for (int number = 33; number <= 126; number++)  {  if (element == number)  {  sorted\_random\_array[index] = number;  }  }  }  int flag\_for\_file = 0;  transferring\_data\_to\_file(flag\_for\_file, count\_elements, sorted\_random\_array);  // Выбираем тип сортировки  answer = type\_of\_sort();  // Передаём данные в выбранную сортировку  if (answer == 1)  {  // Сортировка пузырьком  bubble\_sort(count\_elements, sorted\_random\_array);    answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  else if (answer == 2)  {  // Сортировка вставкой  sorting\_by\_insertion(count\_elements, sorted\_random\_array);  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  else if(answer == 3)  {  // Быстрая сортировка  clock\_t time\_start= clock();  quick\_sort(sorted\_random\_array, 0, count\_elements - 1);  clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, (count\_elements \* 2), sorted\_random\_array);  // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью быстрой сортировки:\n{");  for (int index = 0; index < count\_elements; index++)  {  for (int number = 33; number <= 126; number++)  if (sorted\_random\_array[index] == number)  {  printf("%c", sorted\_random\_array[index]);  }  }  printf("}\n");  printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  }  // 2.Ввести символы самостоятельно.  case 2:  {  // Основные данные для функционала программы  printf("\nВведите кол-во символов, которые хотите ввести: ");  scanf("%d", &count\_elements);  printf("Введите символы через Enter.\n");    // Массив, в который будут передаваться данные пользователя  int array\_of\_characters[count\_elements \* 2];  int sorted\_array[count\_elements \* 2];  // Переменная для считывания символа с клавиатуры  char element;  // Считываем символы с клавиатуры  for (int index = 0; index < (count\_elements \* 2); index++)  {  scanf("%c", &element);  array\_of\_characters[index] = element;  }    // Убираем из массива ненужные символы  int index\_in\_array = 0;  for (int index = 0; index < (count\_elements \* 2); index++)  {  int element;  element = (int)array\_of\_characters[index];  for (int number = 33; number <= 126; number++)  {  if (element == number)  {  sorted\_array[index\_in\_array] = number;  index\_in\_array++;  }  }  }  int flag\_for\_file = 0;  transferring\_data\_to\_file(flag\_for\_file, count\_elements, sorted\_array);  // Выбираем тип сортировки  answer = type\_of\_sort();  // Передаём данные в выбранную сортировку  if (answer == 1)  {  // Сортировка пузырьком  bubble\_sort(count\_elements, sorted\_array);  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  else if (answer == 2)  {  // Сортировка вставкой  sorting\_by\_insertion(count\_elements, sorted\_array);  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  else if(answer == 3)  {  // Быстрая сортировка  clock\_t time\_start= clock();  quick\_sort(sorted\_array, 0, (count\_elements \* 2) - 1);    clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, (count\_elements \* 2), sorted\_array);  // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью быстрой сортировки:\n{");  for (int index = 0; index < (count\_elements \* 2); index++)  {  for (int number = 33; number <= 126; number++)  if (sorted\_array[index] == number)  {  printf("%c", sorted\_array[index]);  }  }  printf("}\n");    printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);  answer = launching\_the\_program();  if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  }  default:  {  printf("Неверный ввод, введите 1 или 2.\n");  return main();  }  }  return 0;  } |

1. **Вторая программа.**

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <stdbool.h>  #include <time.h>  // Выбираем сортировку  int type\_of\_sort()  {  int answer;  printf("\nВыберите тип сортировки символов:\n");  printf("1.Сортировка пузырьком.\n");  printf("2.Сортировка вставкой.\n");  printf("3.Быстрая сортировка.\n");  printf("Ваш ответ: ");  scanf("%d", &answer);  if (answer == 1 || answer == 2 || answer == 3)  {  return answer;  }  else  {  printf("Неверный ввод, введите 1 или 2, или 3.\n");  return type\_of\_sort();  }  }  // Достаём данные из файла  int output\_data\_from\_file(int count, int array[])  {  int answer;  printf("Хотите вывести на экран все символы из файла?\n");  printf("1.Да.\n2.Нет.\nВаш ответ: ");  scanf("%d", &answer);  printf("\n");  if (answer == 1)  {  printf("Символы, которые находятся в файле:\n{");  for (int index = 0; index < count; index++)  {  for (int number = 33; number <= 126; number++)  {  if (array[index] == number)  {  printf("%c", array[index]);  }  }  }  printf("}\n");  }  else if(answer != 1 && answer != 2)  {  printf("Неверный ввод.\nОтветьте на вопрос заново.\n\n");  return output\_data\_from\_file(count, array);  }  return 0;  }  /\* Передаём данные с массива в новый файл для сохранения д  анных после сортировки \*/  int transferring\_data\_to\_file(int flag, int count, int array[])  {  FILE \*file;  if (flag == 0)  {  file = fopen("Sortng.txt", "w");  fprintf(file, "Неотсортированный массив:\n");  for( int index = 0; index < count; index++)  {  for (int number = 33; number <= 126; number++)  if (array[index] == number)  {  fprintf(file, "%c ", array[index]);  }  }  fprintf(file, "\n");  fclose(file);  }  else if (flag == 1)  {  file = fopen("Sortng.txt", "a+");  fprintf(file, "Отсортированный массив:\n");  for( int index = 0; index < count ; index++)  {  for (int number = 33; number <= 126; number++)  {  if (array[index] == number)  {  fprintf(file, "%c ", array[index]);  }  }  }  fclose(file);  }  return 0;  }  // Сортировка пузырьком  int bubble\_sort(int count, int array[])  {  int element;  bool flag;  clock\_t time\_start= clock();  for (int circle = count - 1; circle >= 0; circle--)  {  flag = 1;  for (int index = 0; index < circle; index++)  {  if (array[index] > array[index + 1])  {  element = array[index];  array[index] = array[index + 1];  array[index + 1] = element;  flag = 0;  }  }  if (flag == 1)  {  break;  }  }  clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, (count \* 2), array);    // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью сортировки пузырьком:\n{");  for (int index = 0; index < count; index++)  {  for (int number = 33; number <= 126; number++)  if ((int)array[index] == number)  {  printf("%c", array[index]);  }  }  printf("}\n");    printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);  return 0;  }  // Сортировка вставкой  int sorting\_by\_insertion(int count, int array[])  {  int element, location;  clock\_t time\_start= clock();    for (int index = 1; index < count; index++)  {  element = array[index];  location = index - 1;  while(location >= 0 && array[location] > element)  {  array[location+1] = array[location];  location = location - 1;  }  array[location+1] = element;  }  clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, count, array);  // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью сортировки вставкой:\n{");  for (int index = 0; index < count; index++)  {  for (int number = 33; number <= 126; number++)  if ((int)array[index] == number)  {  printf("%c", array[index]);  }  }  printf("}\n");    printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);  return 0;  }  // Быстрая сортирровка  int quick\_sort(int array[], int begin, int end)  {  if (begin < end)  {  int left = begin, right = end, middle = array[(left + right) / 2];  do  {  while (array[left] < middle)  {  left++;  }  while (array[right] > middle)  {  right--;  }  if (left <= right)  {  int tmp = array[left];  array[left] = array[right];  array[right] = tmp;  left++;  right--;  }  }  while (left <= right);  {  quick\_sort(array, begin, right);  quick\_sort(array, left, end);  }  }  return 0;  }  // Запрос на сброс ортировки  int reset\_sorting()  {  int answer;  printf("\nХотите сбросить сортировку?\n");  printf("1.Да.\n2.Нет.\nВаш ответ: ");  scanf("%d", &answer);  printf("\n");  if (answer == 1 || answer == 2)  {  return answer;  }  else  {  printf("Неверный ввод, введите 1 или 2.\n");  return reset\_sorting();  }  }  // Запуск программы по новой  int launching\_the\_program()  {  int answer;  printf("\nХотите повторить запуск программы по новой для сортировки массива?\n");  printf("1.Да.\n2.Нет.\nВаш ответ: ");  scanf("%d", &answer);  printf("\n");  if (answer == 1 || answer == 2)  {  return answer;  }  else  {  printf("Неверный ввод, введите 1 или 2.\n");  return launching\_the\_program();  }  }  // Основа  int main()  {  // Открываем файл  FILE \*file;  file = fopen("text.txt", "r");  // Осноные переменные  int answer;  int count\_elements, index\_in\_array = 0;  // Считываем кол-во элементов в файле  fscanf(file, "%d", &count\_elements);  int array\_of\_characters[count\_elements \* 2];  // Считываем символы из файла и добавляем их в массив  for (int index = 0; index < (count\_elements \* 2); index++)  {  char element;  fscanf(file, "%c", &element);  for (int number = 33; number <= 126; number++)  {  if (element == number)  {  array\_of\_characters[index\_in\_array] = number;  index\_in\_array++;  }  }  }  output\_data\_from\_file(count\_elements, array\_of\_characters);  answer = type\_of\_sort();  int flag\_for\_file = 0;  transferring\_data\_to\_file(flag\_for\_file, count\_elements, array\_of\_characters);  switch (answer)  {  case 1:  {  // Сортировка пузырьком  bubble\_sort(count\_elements, array\_of\_characters);  printf("\nСортировка прошла успешно.\nОтсортированные символы находятся в файле 'Sorting.txt'.\n");  answer = reset\_sorting();  if (answer == 1)  {  printf("Сортировка сброшена.Программа запустится по новой, ожидайте.\n\n");  return main();  }  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  case 2:  {  // Сортировка вставкой  sorting\_by\_insertion(count\_elements, array\_of\_characters);  printf("\nСортировка прошла успешно.\nОтсортированные символы находятся в файле 'Sorting.txt'.\n");  answer = reset\_sorting();  if (answer == 1)  {  printf("Сортировка сброшена.Программа запустится по новой, ожидайте.\n\n");  return main();  }  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  case 3:  {  // Быстрая сортировка  clock\_t time\_start= clock();  quick\_sort(array\_of\_characters, 0, count\_elements - 1);  clock\_t time\_end = clock() - time\_start;  // Передаём отсортированный массив в файл  int flag\_for\_file = 1;  transferring\_data\_to\_file(flag\_for\_file, (count\_elements \* 2), array\_of\_characters);  // Выводим получившийся массив после сортировки  printf("\nНовый массив, отсортированный с помощью быстрой сортировки:\n{");  for (int index = 0; index < count\_elements; index++)  {  for (int number = 33; number <= 126; number++)  if (array\_of\_characters[index] == number)  {  printf("%c", array\_of\_characters[index]);  }  }  printf("}\n");  printf("Время сортировки: %lf\n", (double)time\_end / CLOCKS\_PER\_SEC);    printf("\nСортировка прошла успешно.\nОтсортированные символы находятся в файле 'Sorting.txt'.\n");    answer = reset\_sorting();  if (answer == 1)  {  printf("Сортировка сброшена.Программа запустится по новой, ожидайте.\n\n");  return main();  }  answer = launching\_the\_program();    if (answer == 1)  {  return main();  }  else if (answer == 2)  {  printf("Программа остановлена.\n");  break;  }  }  }    return 0;  } |