МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проектированию

По курсу «Логика и основы алгоритмизации в инженерных задачах» на тему «Реализация алгоритма генерирования всех перестановок заданного множества в лексикографическом порядке»

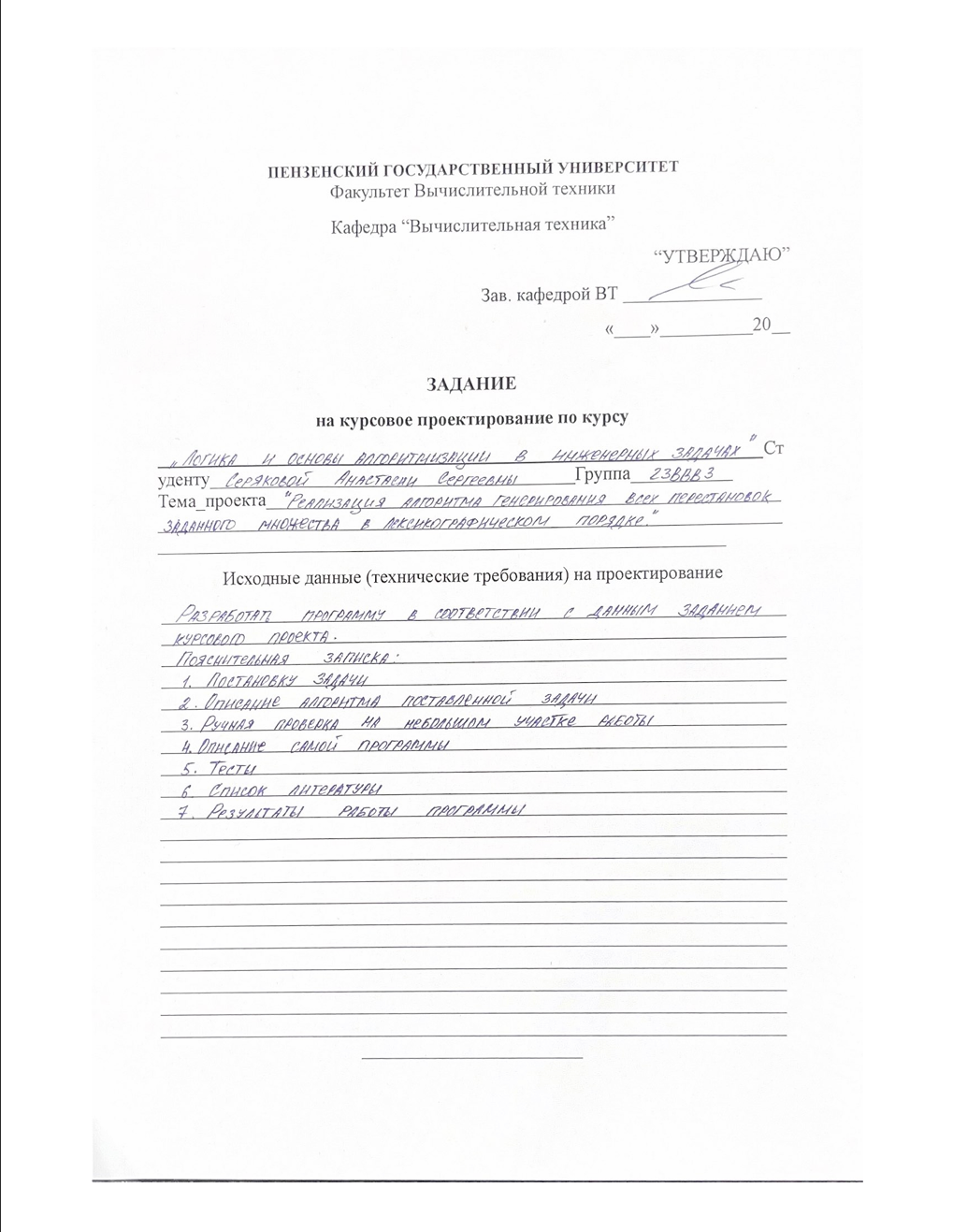
Выполнил студент группы 23ВВВ3:

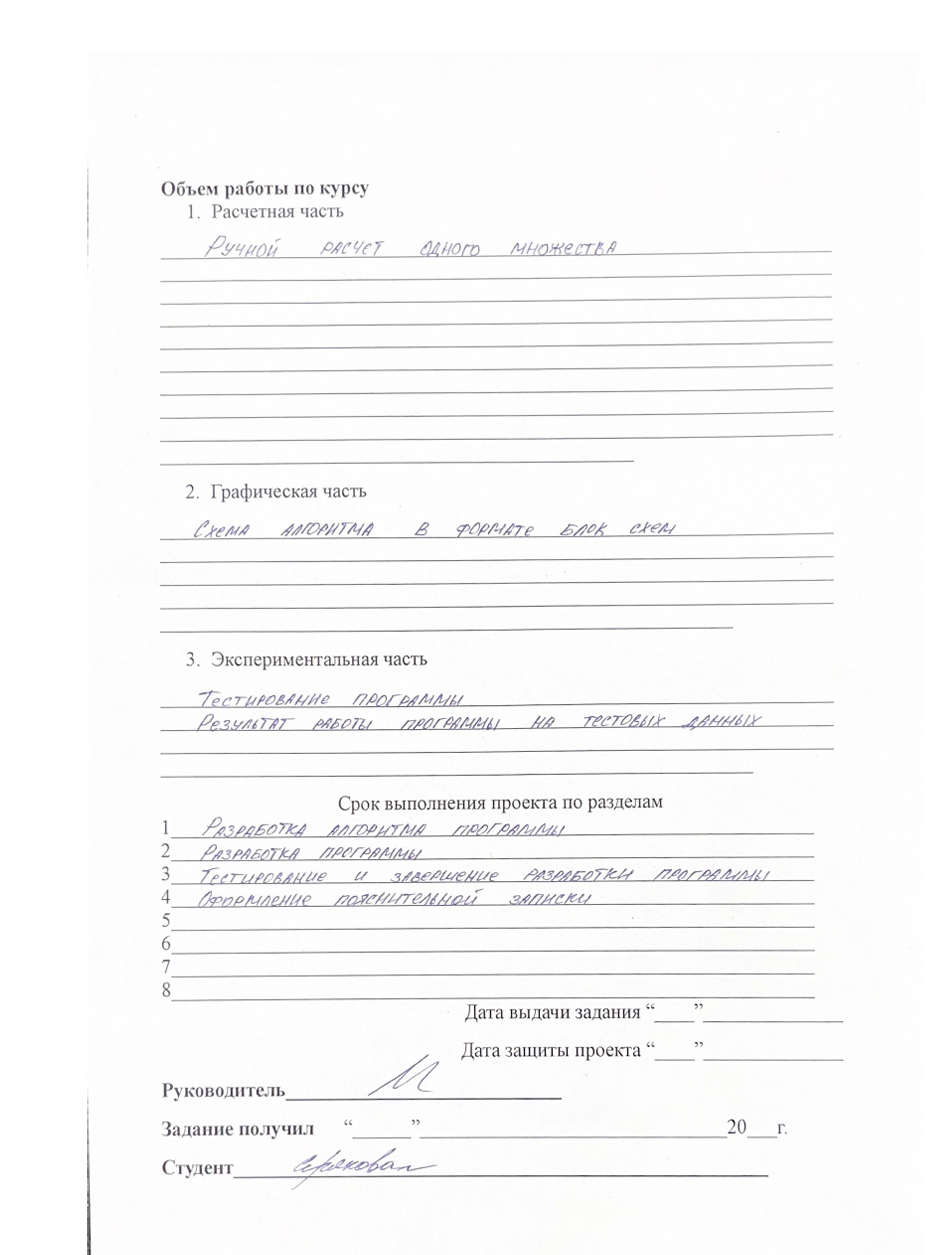
Серякова А.С.

Принял:

Митрохин М.А.

Пенза, 2024

****

****

**Содержание**

[Реферат 5](#_Toc183589969)

[Введение 6](#_Toc183589970)

[Постановка задачи 8](#_Toc183589971)

[Описание алгоритма программы 9](#_Toc183589972)

[Описание программы 12](#_Toc183589973)

[Тестирование 15](#_Toc183589974)

[Ручной расчёт задачи 19](#_Toc183589975)

[Заключение 20](#_Toc183589976)

[Список литературы 21](#_Toc183589977)

[Приложение А. 22](#_Toc183589978)

# Реферат

Отчет 23 стр, 10 рисунков, 3 таблица.

ЛЕКСИКОГРАФИЧЕСКИЙ ПОРЯДОК.

Цель исследования – разработка программы, способная выводить все возможные перестановки множества в лексикографическом порядке.

Программа принимает на вход множество символов (строку), которое пользователь может ввести вручную или сгенерировать случайным образом, указав его длину. Затем программа находит все перестановки символов этого множества и сортирует их в лексикографическом порядке (алфавитный порядок). После сортировки программа выводит только те перестановки, которые предшествуют исходному множеству в лексикографическом порядке. Если таких перестановок нет, то выводится соответствующее сообщение. Программа также предоставляет пользователю возможность сохранить результаты в файл.

# Введение

Данная курсовая работа посвящена разработке программы, которая генерирует все возможные перестановки символов в заданной строке в лексикографическом порядке. Программа позволяет вводить строку вручную или генерировать случайную строку заданной длины. Результаты работы программы могут быть выведены на экран и/или сохранены в текстовый файл по выбору пользователя. Работа основана на модифицированном алгоритме генерации перестановок, обеспечивающем вывод в лексикографическом порядке.

Программа, генерирующая перестановки в лексикографическом порядке и выводящая только те, которые меньше исходного множества, может быть полезна в нескольких областях:

1. Комбинаторная оптимизация: В задачах комбинаторной оптимизации часто требуется перебрать все возможные комбинации или перестановки и выбрать наилучшую. Лексикографический порядок может быть полезен для эффективного перебора вариантов, особенно если есть критерий упорядочения, позволяющий отбросить заведомо худшие варианты (те, которые больше исходного множества). Например, поиск оптимального порядка выполнения операций в задаче планирования.
2. Генерация паролей: Программа может быть использована для генерации паролей, удовлетворяющих определенным критериям. Если есть набор символов и требуется сгенерировать пароли, которые меньше заданного пароля (в лексикографическом порядке), программа может помочь в этом.
3. Тестирование и отладка: В контексте тестирования программного обеспечения программа может быть использована для генерации входных данных для тестирования алгоритмов сортировки или поиска.
4. Криптография (в ограниченных случаях): В некоторых криптографических алгоритмах может потребоваться перебор перестановок, и лексикографический порядок может быть полезен для упорядочивания и фильтрации результатов. Однако, для большинства криптографических задач потребуются значительно более сложные и безопасные алгоритмы.
5. Математические исследования: Программа может быть использована в математических исследованиях, связанных с комбинаторикой и теорией перестановки.
6. Образовательные цели: Программа может служить наглядным примером работы с перестановками, сортировкой и алгоритмами.

Для реализации проекта выбран язык программирования C++. Этот выбор обусловлен рядом преимуществ:

1. Высокая производительность: C++ — компилируемый язык, что обеспечивает высокую скорость выполнения кода, особенно важную при работе с рекурсивными алгоритмами, как в данной задаче. Генерация всех перестановок для длинных строк может потребовать значительных вычислительных ресурсов, и C++ позволяет эффективно справляться с этой задачей.
2. Поддержка объектно-ориентированного программирования (ООП): Хотя в данном проекте ООП используется в ограниченном объеме, C++ позволяет структурировать код более эффективно, что упрощает разработку, тестирование и последующее обслуживание программы.
3. Удобство работы со стандартной библиотекой: Стандартная библиотека C++ (iostream, fstream, string) предоставляет удобные инструменты для ввода-вывода, работы с файлами и обработки строк, что существенно упрощает разработку программы.
4. Широкая распространенность и поддержка: C++ — широко используемый язык программирования с большим сообществом и обширной документацией, что облегчает поиск решений и получение помощи при необходимости.

# Постановка задачи

Необходимо разработать консольное приложение на языке C++, которое выполняет следующие действия:

1. Предоставляет пользователю выбор ввода строки: вручную или с помощью генератора случайных строк.
2. Генерирует все возможные перестановки символов введенной строки в лексикографическом порядке, используя рекурсивный алгоритм.
3. Выводит сгенерированные перестановки на экран.
4. Предоставляет пользователю возможность сохранить все сгенерированные перестановки в текстовый файл.

# Описание алгоритма программы

Для программной реализации алгоритма понадобится 2 функции: *permute* - рекурсивная функция, которая генерирует все перестановки строки, *generateRandomString* - генерирует случайную строку заданной длины.

Для начала пользователю предлагается выбраться:

1. Ввести строку вручную.
2. Задать строку случайно, выбрав её размер, в этом поможет нам функция *generateRandomString*

Программа запрашивает у пользователя способ ввода строки: вручную или сгенерировать случайную. В случае ручного ввода, программа ждет ввод строки от пользователя. В случае генерации случайной строки, программа запрашивает длину строки и генерирует случайную строку заданной длины из символов латинского алфавита (в верхнем и нижнем регистре) и цифр.

Функция *permute* рекурсивно генерирует все возможные перестановки символов в строке. Она работает путем последовательного обмена символов в строке. Сгенерированные перестановки сортируются в лексографическом порядке (от A до Z, от 0 до 9) с помощью стандартной функции *sort*. Отсортированные перестановки проходят проверку, которые ниже нашего заданного множетсва, тогда они выводятся на экран.

В заключение пользователю даётся возможность сохранить результаты работы программы в файл, обрабатывая при этом возможные ошибки и освобождая используемую память.

Ниже представлен псевдокод всех функций, а так же *main*.

Функция permute(строка str, целое число l, целое число r, список строк result):

ЕСЛИ l равно r ТО:

Добавить str в конец списка result

ИНАЧЕ:

Для каждого целого числа i от l до r:

Поменять местами str[l] и str[i]

Вызвать permute(str, l + 1, r, result) // Рекурсивный вызов

Поменять местами str[l] и str[i] // Вернуть в исходное состояние (важно!)

Конец цикла

Конец ЕСЛИ

Конец функции

Процедура main():

Начать цикл:

Показать меню выбора ввода строки (вручную или случайная генерация).

Получить выбор пользователя.

Обработать некорректный ввод.

Если выбор — выход, закончить цикл.

Если выбор — ручной ввод:

Получить строку от пользователя.

Если выбор — случайная генерация:

Получить длину строки от пользователя.

Обработать некорректный ввод длины.

Сгенерировать случайную строку указанной длины.

Если строка не пуста:

Сгенерировать все перестановки строки.

Отсортировать перестановки в лексикографическом порядке.

Вывести перестановки, которые меньше исходной строки.

Если таких перестановок нет, вывести сообщение об этом.

Спросить пользователя, сохранить ли результаты в файл.

Если пользователь хочет сохранить:

Получить имя файла от пользователя.

Открыть файл для записи.

Если файл открыт:

Записать отфильтрованные перестановки в файл.

Закрыть файл.

Вывести сообщение об успешном сохранении.

Иначе:

Вывести сообщение об ошибке.

Конец цикла

Конец процедуры

# Описание программы

Для написания данной программы использован язык программирования Си.Язык - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

Данная программа является многомодульной, поскольку состоит из нескольких функций.

Работа программы начинается с запроса генерации множества. Если пользователь выбрал сгенерировать случайно, то на экран выводится запрос на количество символов. В ином случае пользователь сам вводит строку и подтверждает.

Работа начинается с запроса выбора способа ввода строки:

cout << "Выберите способ ввода строки:\n";

cout << "1. Ввести строку вручную\n";

cout << "2. Сгенерировать случайную строку\n";

cout << "0. Выход\n";

cout << "Ваш выбор: ";

cin >> choice;

Если пользователь выбрал генерацию случайной строки, программа запрашивает длину и генерирует строку:

if (cin.fail()) {

cout << "Некорректный ввод. Попробуйте снова.\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

continue;

}

Произведение перестановок и их вывод:

if (choice == 2) {

int length;

cout << "Введите длину случайной строки: ";

cin >> length;

}

Пользователь выбирает, сохранять ли результаты в файл:

vector<string> permutations;

permute(inputString, 0, inputString.length() - 1, permutations);

sort(permutations.begin(), permutations.end(), greater<string>());

char saveChoice;

cout << "Сохранить результаты в файл? (y/n): ";

cin >> saveChoice;

Если выбран «y», программа просит название файла и сохраняет перестановки:

ofstream outfile(filename);

if (outfile.is\_open()) {

// Запись перестановок

}

Ниже можно увидеть оформление функционала и работа с ним.

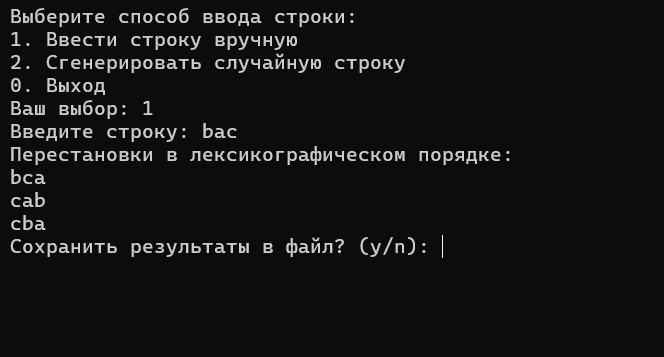


Рисунок 1 – Множество вписано вручную

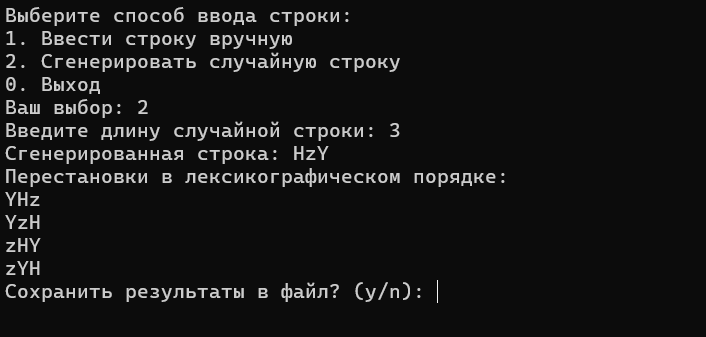


Рисунок 2 - Множество сгенерировано

**Тестирование**

Среда разработки Microsoft Visual Studio 2010 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Тесты:

1. Работа программы при случайном заданном множестве.
   1. Вывод сообщения о выборе ввода строчки:

A black background with white text

Description automatically generated

Рисунок 3 – Запуск программы

* 1. Ввод длины для случайного множества и вывод результата на экран:

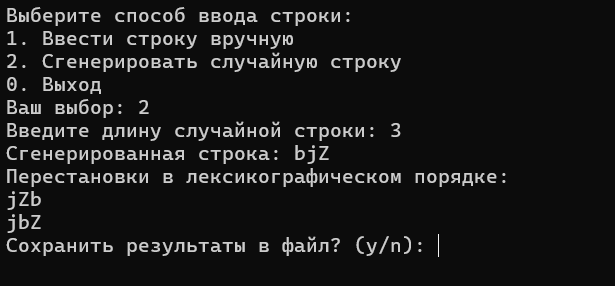


Рисунок 4 – Тестирование программы при случайном множестве

* 1. Сохранение результат в файл:

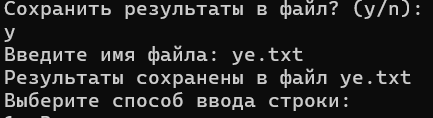


Рисунок 5 – Сохранение результата в файл



Рисунок 6 – Результат в txt файле

Таблица 1 – Тест 1.

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод сообщения о выборе:  Сгенерировать множество или ввести вручную | Верно |
| Сгенерировать случайное множество | Вывод сообщения о количестве символов в множестве | Верно |
| Сохранение результата | Сохранение результат в файл который вписал пользователь | Верно |

1. Работа программы при вводе множества в ручную.
2. Вывод сообщения о выборе ввода строчки:

A black background with white text

Description automatically generated

Рисунок 7 – Запуск программы

1. Ввод множества вручную и вывод результата в консоль:

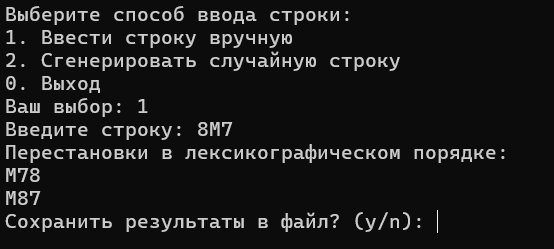


Рисунок 8 – Результат работы с введенным множеством

1. Сохранение результата в файл:

A black background with white text

Description automatically generated

Рисунок 9 – Сохранение в файл при ручном вводе множества



Рисунок 10 – Результат в txt файле.

Таблица 2 – Тест 2.

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод сообщения о выборе:  Сгенерировать множество или ввести вручную | Верно |
| Ввод множества вручную | Вывод всех перестановок в лексикографическом порядке | Верно |
| Сохранение результата | Сохранение результат в файл который вписал пользователь | Верно |

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимых требований.

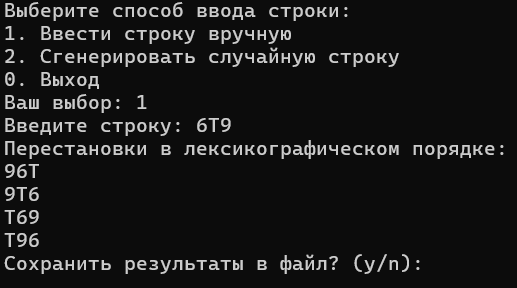
# Ручной расчёт задачи

Перестановки символов “6Т9” в лексикографическом порядке (от меньшего к большему):

Распишем все перестановки данного множества. Это будут: “69T”, “6T9”, “96T”, “9T6”, “T69”, “T96”. Далее сортируем их в лексикографическом порядке - “69T”, “6T9”, “96T”, “9T6”, “T69”, “T96”.

После того, как мы сделали их от меньшего к большему, сравниваем исходное множество с каждой перестановкой и выводим только те, которые меньше “6T9”: “96T”, “9T6”, “T69”, “T96”.

Так мы получаем все перестановки нашего множества, которые меньше “6T9”. Из нашего ручного расчёта мы можем сделать вывод, что наша программа работает правильно, что доказывается на рисунке ниже

.

# Заключение

В ходе работы было реализовано создание программного продукта, позволяющего наглядно представить все возможные варианты перестановок элементов множества. Одна из главных особенностей этого программного продукта - представление результата перестановок в лексикографическом порядке. Поставленная задача была полностью реализована.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса. Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

1. Новиков Ф.А. «Дискретная математика для программистов» 2-е изд. – СПб.: Питер, 2000.
2. Герберт Шилдт «Полный справочник по C++» - Вильямс, 2006 5. Х. М. Дейтел, П. Дж. Дейтел «Как программировать на C++» - Бином-Пресс, 2008.

# Приложение А.

Листинг программы

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <fstream>

using namespace std;

// Функция генерации перестановок (лексикографический порядок)

void permute(string str, int l, int r, vector<string>& result) {

if (l == r) {

result.push\_back(str);

}

else {

for (int i = l; i <= r; i++) {

swap(str[l], str[i]);

permute(str, l + 1, r, result);

swap(str[l], str[i]);

}

}

}

// Функция генерации случайной строки

void generateRandomString(string& str, int length) {

const string charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

for (int i = 0; i < length; ++i) {

str += charset[rand() % charset.length()];

}

}

int main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

string inputString;

int choice;

do {

cout << "Выберите способ ввода строки:\n";

cout << "1. Ввести строку вручную\n";

cout << "2. Сгенерировать случайную строку\n";

cout << "0. Выход\n";

cout << "Ваш выбор: ";

cin >> choice;

//Проверка на корректность ввода

if (cin.fail()) {

cout << "Некорректный ввод. Попробуйте снова.\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

continue;

}

if (choice == 0) break;

if (choice == 1) {

cout << "Введите строку: ";

cin.ignore();

getline(cin, inputString);

}

else if (choice == 2) {

int length;

cout << "Введите длину случайной строки: ";

cin >> length;

if (cin.fail() || length <= 0) {

cout << "Некорректная длина строки. Попробуйте снова.\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

continue;

}

inputString = "";

generateRandomString(inputString, length);

cout << "Сгенерированная строка: " << inputString << endl;

}

else {

cout << "Некорректный выбор. Попробуйте снова.\n";

}

if (!inputString.empty()) {

vector<string> permutations;

permute(inputString, 0, inputString.length() - 1, permutations);

sort(permutations.begin(), permutations.end());

cout << "Перестановки в лексикографическом порядке:\n";

for (const string& p : permutations) {

if (p > inputString) {

cout << p << endl;

}

}

char saveChoice;

cout << "Сохранить результаты в файл? (y/n): ";

cin >> saveChoice;

cin.ignore();

if (tolower(saveChoice) == 'y') {

string filename;

cout << "Введите имя файла: ";

getline(cin, filename);

ofstream outfile(filename);

if (outfile.is\_open()) {

for (const string& p : permutations) {

if (p > inputString) {

outfile << p << endl;

}

}

outfile.close();

cout << "Результаты сохранены в файл " << filename << endl;

}

else {

cerr << "Ошибка при открытии файла!" << endl;

}

}

permutations.clear();

}

} while (choice != 0);

return 0;

}