*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № 7**

**Название лабораторной работы: Программирование с использованием контейнеров STL и Qt**

**Дисциплина: Объектно-ориентированное программирование**

Студент гр. ИУ6-22Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Плютто**



(Подпись, дата) (И.О. Фамилия)



Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Задание

Организовать программным способом символьные файлы ***F*** и ***G.*** С помощью множеств stl::set и QSet определить, совпадают ли слова в этих файлах. Выдать список слов, которые не совпадают. Реализовать свои функции сравнения, игнорирующие различия в регистре символов. Сравнить скорость работы разных реализаций в отладочном и финальном режиме.

Решение

1. Создадим функция по генерации слов

1. QString generateWords()

2. {

3. srand(time(NULL));

4. char alphabet[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

5. QString word = "";

6. for (int j = 0; j < arc4random() % 10000; j++)

7. {

8. int length = arc4random() % 10 + 1;

9. for (int i = 0; i < length; i++)

10. {

11. int index = arc4random() % 26;

12. char letter = alphabet[index];

13. word += letter;

14. }

15. word += ' ';

16. }

17. return word;

18. }

19.

1. Для предыдущей функции создадим функцию записи сгенерированных слов в файл

1. void dofiles()

2. {

3. QFile fileF("F");

4. QFile fileG("G");

5. if (fileF.open(QIODevice::WriteOnly | QIODevice::Text) &&

6. fileG.open(QIODevice::WriteOnly | QIODevice::Text))

7. {

8. QTextStream writeStreamF(&fileF);

9. QTextStream writeStreamG(&fileG);

10. writeStreamF << generateWords();

11. writeStreamG << generateWords();

12. fileF.close();

13. fileG.close();

14. }

15. }

16.

1. Создадим функции которые переводят все слова в слова со строчными буквами

1. string toLowercase(string s)

2. {

3. string result = "";

4. for (int i = 0; i < (int)s.length(); i++)

5. result += tolower(s[i]);

6. return result;

7. }

8.

9. QString toLowercase(QString s)

10. {

11. QString result = s.toLower();

12. return result;

13. }

14.

1. Создадим отдельные функции сравнения

1. void comparison(set<string> &fsset, set<string> &gsset)

2. {

3. bool fla = 1;

4. for (const auto &elementF : fsset)

5. {

6. bool fl = 1;

7. for (const auto &elementG : gsset)

8. if (toLowercase(elementF) == toLowercase(elementG))

9. {

10. fl = 0;

11. fla = 0;

12. break;

13. }

14. if (fl)

15. qDebug() << QString::fromStdString(elementF) << " ";

16. }

17. if (fla) qDebug() << "Sets matches";

18. }

19.

20. void comparison(QSet<QString> &fqset, QSet<QString> gqset)

21. {

22. bool fla = 1;

23. for (auto elementF : fqset)

24. {

25. bool fl = 1;

26. for (auto elementG : gqset)

27. if (toLowercase(elementF) == toLowercase(elementG))

28. {

29. fl = 0;

30. fla = 0;

31. break;

32. }

33. if (fl)

34. qDebug() << elementF << " ";

35. }

36. if (fla) qDebug() << "Sets matches";

37. }

38.

Ниже приведен полный код программы и тестирующая функция

1. #include <QCoreApplication>

2. #include <QSet>

3. #include <QString>

4. #include <QFile>

5. #include <set>

6. #include <string>

7. #include <cstdlib>

8. #include <ctime>

9. #include <iostream>

10. #include <chrono>

11. #include <QElapsedTimer>

12. using std::set, std::string, std::rand;

13.

14. // Функция, генерирующая слова

15. QString generateWords()

16. {

17. srand(time(NULL));

18. char alphabet[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

19. QString word = "";

20. for (int j = 0; j < arc4random() % 10000; j++)

21. {

22. int length = arc4random() % 10 + 1;

23. for (int i = 0; i < length; i++)

24. {

25. int index = arc4random() % 52;

26. char letter = alphabet[index];

27. word += letter;

28. }

29. word += ' ';

30. }

31. return word;

32. }

33.

34. // Функции, перобразующие слова с заглавными буквами в слова со всеми прописными

35. string toLowercase(string s)

36. {

37. string result = "";

38. for (int i = 0; i < (int)s.length(); i++)

39. result += tolower(s[i]);

40. return result;

41. }

42.

43. QString toLowercase(QString s)

44. {

45. QString result = s.toLower();

46. return result;

47. }

48.

49. // Функция, генерирующая файлы

50. void dofiles()

51. {

52. QFile fileF("F");

53. QFile fileG("G");

54. if (fileF.open(QIODevice::WriteOnly | QIODevice::Text) &&

55. fileG.open(QIODevice::WriteOnly | QIODevice::Text))

56. {

57. QTextStream writeStreamF(&fileF);

58. QTextStream writeStreamG(&fileG);

59. writeStreamF << generateWords();

60. writeStreamG << generateWords();

61. fileF.close();

62. fileG.close();

63. }

64. }

65.

66. // Функции сравнения

67. void comparison(set<string> &fsset, set<string> &gsset)

68. {

69. bool fla = 1;

70. for (const auto &elementF : fsset)

71. {

72. bool fl = 1;

73. for (const auto &elementG : gsset)

74. if (toLowercase(elementF) == toLowercase(elementG))

75. {

76. fl = 0;

77. fla = 0;

78. break;

79. }

80. if (fl)

81. qDebug() << QString::fromStdString(elementF) << " ";

82. }

83. if (fla) qDebug() << "Sets matches";

84. }

85.

86. void comparison(QSet<QString> &fqset, QSet<QString> gqset)

87. {

88. bool fla = 1;

89. for (auto elementF : fqset)

90. {

91. bool fl = 1;

92. for (auto elementG : gqset)

93. if (toLowercase(elementF) == toLowercase(elementG))

94. {

95. fl = 0;

96. fla = 0;

97. break;

98. }

99. if (fl)

100. qDebug() << elementF << " ";

101. }

102. if (fla) qDebug() << "Sets matches";

103. }

104.

105. int main(int argc, char \*argv[])

106. {

107. QCoreApplication a(argc, argv);

108. set<string> fsset;

109. set<string> gsset;

110. QSet<QString> fqset;

111. QSet<QString> gqset;

112. dofiles();

113. QFile fileF("F");

114. QFile fileG("G");

115. if (fileF.open(QIODevice::ReadOnly | QIODevice::Text) &&

116. fileG.open(QIODevice::ReadOnly | QIODevice::Text))

117. {

118. QTextStream readStreamF(&fileF);

119. QTextStream readStreamG(&fileG);

120. QString Qfstr = readStreamF.readLine();

121. QString Qgstr = readStreamG.readLine();

122.

123. string fstr = Qfstr.toStdString();

124. string gstr = Qgstr.toStdString();

125.

126. // Начало реализации с помощью stl::set

127.

128. QElapsedTimer timer1;

129. timer1.start();

130.

131. // Добавляем все слова из файла F в stl::set

132.

133. int last{};

134. for (int i = 0; i < fstr.length(); i++)

135. {

136. if (fstr[i] == ' ')

137. {

138. fsset.insert(fstr.substr(last + 1, i - last - 1));

139. last = i;

140. };

141. }

142.

143. // Добавляем все слова из файла G в stl::set

144.

145. last = 0;

146. for (int i = 0; i < gstr.length(); i++)

147. {

148. if (fstr[i] == ' ')

149. {

150. gsset.insert(gstr.substr(last + 1, i - last - 1));

151. last = i;

152. };

153. }

154.

155. comparison(fsset, gsset);

156.

157. int t1 = timer1.nsecsElapsed();

158.

159. // Начало реализации с помощью QSet

160.

161. QElapsedTimer timer2;

162. timer2.start();

163.

164. // Добавляем все слова из файла F в QSet

165.

166. last = 0;

167. for (int i = 0; i < Qfstr.length(); i++)

168. {

169. if (fstr[i] == ' ')

170. {

171. fqset.insert(Qfstr.mid(last + 1, i - last - 1));

172. last = i;

173. };

174. }

175.

176. // Добавляем все слова из файла G в QSet

177.

178. last = 0;

179. for (int i = 0; i < Qgstr.length(); i++)

180. {

181. if (fstr[i] == ' ')

182. {

183. gqset.insert(Qgstr.mid(last + 1, i - last - 1));

184. last = i;

185. };

186. };

187. qDebug() << '\n';

188.

189. comparison(fqset, gqset);

190.

191. // Выводим значения с таймера

192.

193. qDebug() << t1 << " ns " << timer2.nsecsElapsed() << " ns";

194. }

195. fileF.close();

196. fileG.close();

197.

198. return a.exec();

199. }

200.

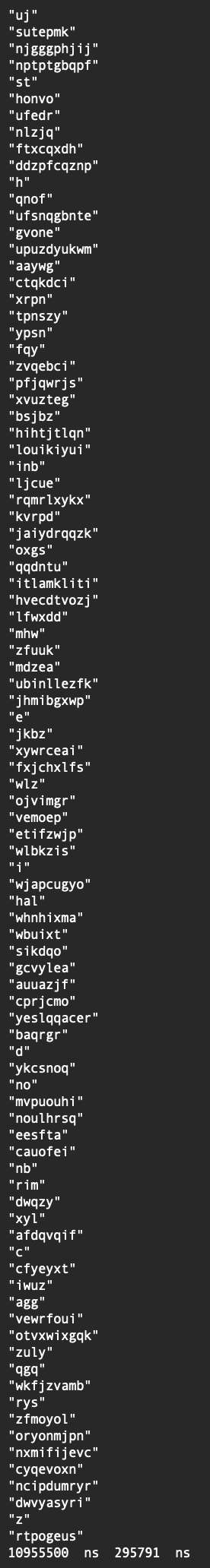
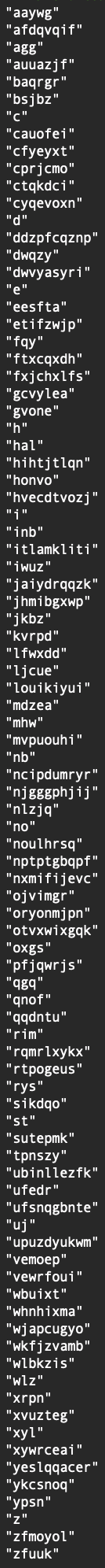


Рисунок : Вывод программы

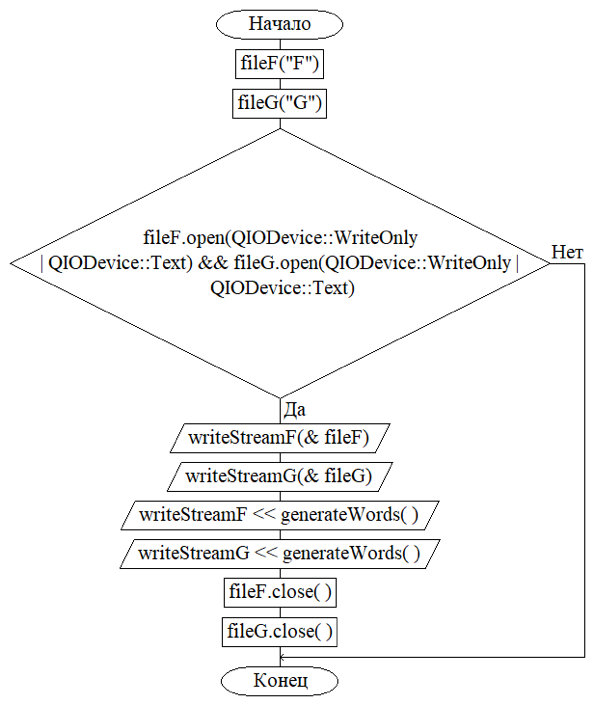


Рисунок : Схема алгоритма для функции, добавляющей в файл

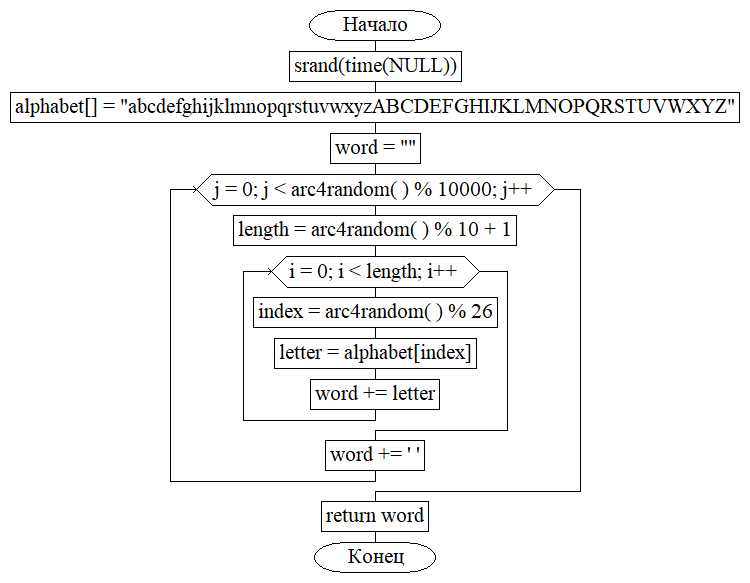


Рисунок Схема алгоритма для функции, создающей слова

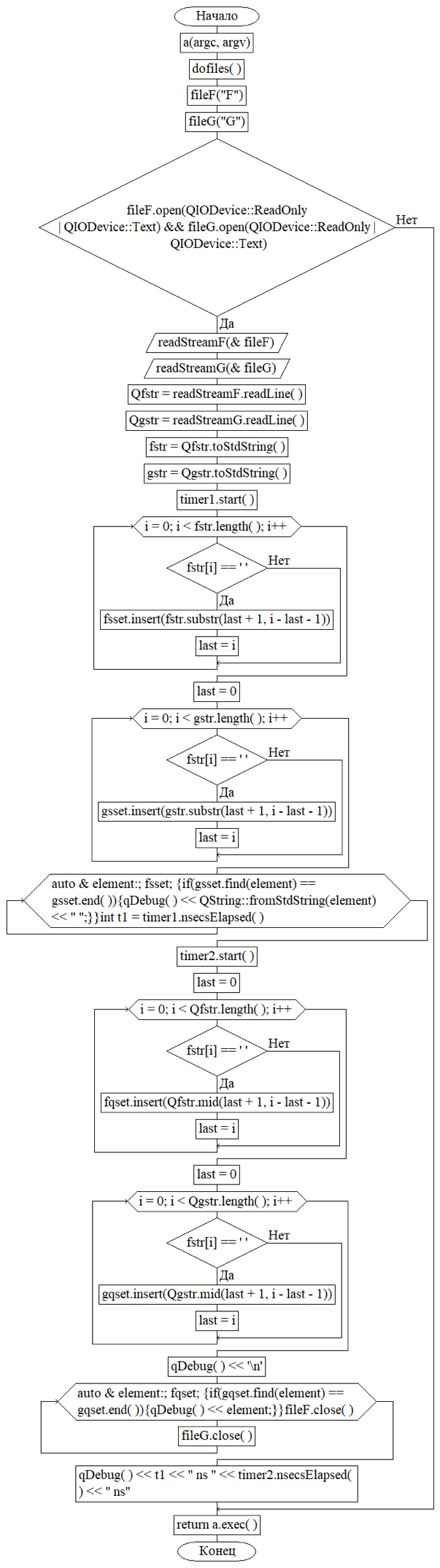


Рисунок : Схема тестирующей программы

Вывод: был изучен и применен на практике контейнер set (в двух разных реализациях: QSet и stl::set). Была написана программа, которая создает базу данных, представленную в виде двух файлов, а также позволяет осуществлять сравнение данных в ней. Были проведены измерения времени для каждой реализации. Результаты этих измерений показывают, что stl::set работает медленнее, чем QSet (предположительно из-за сортировки при добавлении элементов в stl::set).