**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление 15.03.04 «Автоматизация технологических процессов и производств»

Отделение автоматизации и робототехники

Работа со строками в C++

Наименование лабораторной работы

Отчет по лабораторной работе № 4

по дисциплине «Основы программирования и алгоритмизации»

Наименование учебной дисциплины

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент гр. 8Т32 |  | C:\Users\pavlu\OneDrive\Рабочий стол\Подпись.jpg |  |  |  | П. П. Гаврильев |
|  |  | Подпись |  | Дата |  | И.О. Фамилия |
|  |  |  |  |  |  |  |
| Проверил ассистент ОАР |  |  |  |  |  | А. В. Павловский |
| Должность |  | Подпись |  | Дата |  | И.О. Фамилия |

Томск – 2024 г.

**Цель работы:**

Ознакомиться и реализовать решение для работы со строками на практике на языке программирования С++.

**Ход работы:**

Таблица 1 – Формулировка заданий

|  |  |
| --- | --- |
| **№** | Формулировка |
| **1** | В данной строке найти количество цифр. |
| **2** | Дан текст. Найти слова, состоящие из цифр, и сумму чисел, которые образуют эти слова. |
| **3** | Даны две строки. Определите, можно ли из некоторых символов первой строки и всех символов второй строки составить новую строку, в которой каждый символ встречается ровно два раза. |
| **4** | Дан email в строке. Определить, является ли он корректным (наличие символа @ и точки, наличие не менее двух символов после последней точки и т.д.). |
| **5** | Написать генерацию строк длины 10, причем первые 4 символа - цифры, следующие два символы - различные буквы, следующие 4 символа - нули или единицы, причем одна единица точно присутствует. |
| **6** | Дана строка. Если символы в ней упорядочены по алфавиту, то вывести 'yes', иначе вывести первый символ, нарушающий алфавитный порядок. |
| **7** | Дана строка. Придумать алгоритм шифрования данной строки и дешифрования. |

Задание 1.

В данном задании поставлена задача найти количество цифр в тексте. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках 1.1, 1.2, 1.3 и 1.4. По данным блок схемам были написаны программа, которая представлены в листингах 1.1, 1.2, 1.3 и 1.4. Результат выполнения программы представлен на рисунке 1.5.

Программа будет получать размер файла и по этому размеру он будет создавать динамический переменную типа char. После этого программа получит текст из файла, по которому он будет проходиться по всей длине для поиска цифр. Поиск цифр будет реализована с использованием таблицы ASCII. После нахождения цифры к счетчику будет прибавляться единица. Для реализации поставленной задачи была реализована функция digitcount, также для получения размера файла функция filesize и получения строки read\_file, которая дальше будет использоваться в следующих заданиях. После вызова функции digitcount на консоль пользователя выводится количество цифр в файле.

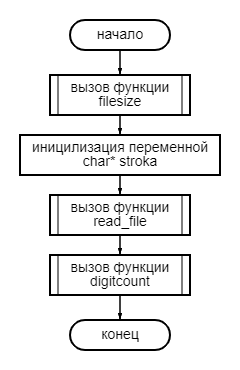


Рисунок 1.1 – Блок схема программы

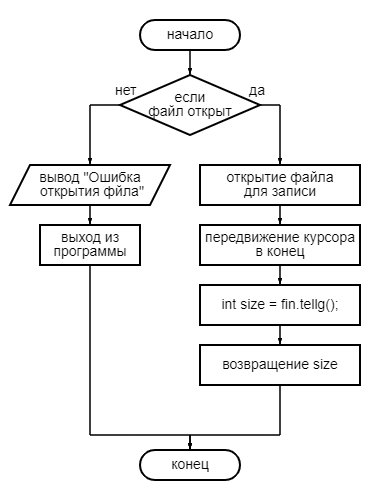


Рисунок 1.2 – Блок схема функции filesize(filename)

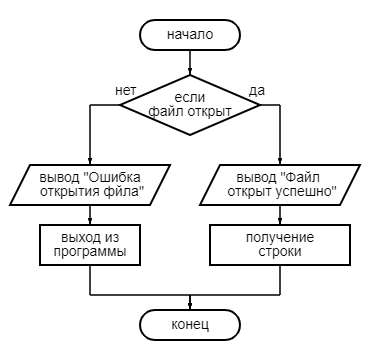


Рисунок 1.3 – Блок схема функции read\_file(filename)

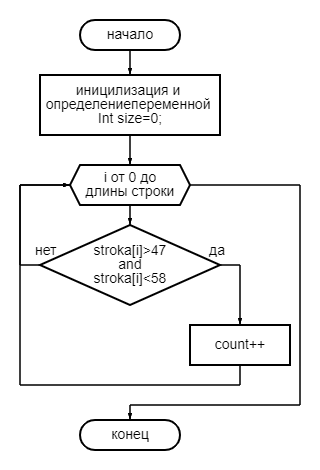


Рисунок 1.4 – Блок схема функции digitcount(stroka)



Рисунок 1.5 – Итог запуска программы

Листинг 1.1 – Код программы

|  |
| --- |
| #include <iostream>  #include <fstream>  #include "read\_file.h"  #include "filesize.h"  #include "digitgount.h"  int main()  {  setlocale(LC\_ALL, "ru");  char\* stroka=new char[filesize("stdfile.txt")];  read\_file(stroka, "stdfile.txt");  digitcount(stroka);  } |

Листинг 1.2 - Библиотека filesize(filename)

|  |
| --- |
| #pragma once  #include <iostream>  #include <fstream>  #include "filesize.h"  int filesize(std::string file\_name)  {  std::ifstream fin(file\_name);  if (fin.is\_open()) {  fin.seekg(0, std::ios::end);  int size = fin.tellg();  return size;  }  else {  std::cout << "Ошибка открытия файла.";  exit(0);  }  fin.close();  } |

Листинг 1.3 - Библиотека read\_file(filename)

|  |
| --- |
| #pragma once  #include <fstream>  #include <iostream>  #include "read\_file.h"  #include "filesize.h"  void read\_file(char\* stroka, std::string file\_name) {  std::ifstream fin(file\_name);  if (fin.is\_open()) {  std::cout << "Файл открыт успешно." << std::endl;  fin.getline(stroka, filesize("stdfile.txt"));  }  else {  std::cout << "Ошибка открытия файла.";  exit(0);  }  fin.close();  } |

Листинг 1.4 - Библиотека digitcount(stroka)

|  |
| --- |
| #pragma once  #include <string>  #include <iostream>  #include "digitgount.h"  void digitcount(char\* stroka)  {  int count = 0;  for (int i = 0; i <strlen(stroka); i++) {  if (stroka[i] > 47 && stroka[i] < 58) {  count++;  }  }  std::cout << "Количество цифр в файле>>" << count;  } |

Задание 2.

В данном задании поставлена задача найти слова, состоящие только из цифр и сумма этих цифр. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках 2.1 и 2.2. По данным блок схемам были написаны программа, которая представлены в листингах 2.1 и 2.2. Результат выполнения программы представлен на рисунке 2.3.

Программа будет получать размер файла и по этому размеру он будет создавать динамический переменную типа char. После этого программа будет получать каждый раз строку из файла пока не дойдет в конец и к каждой строке будет вызывать функцию digitsum. Также для получения размера файла используется функция filesize из первого задания. Поиск цифр будет реализована с использованием таблицы ASCII. После нахождения цифры, они будут суммироваться. После вызова функции digitsum на консоль пользователя выводится слово, состоящее из цифр и его сумма.

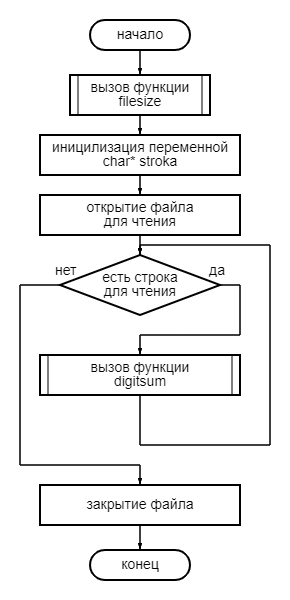


Рисунок 2.1 – Блок схема программы

|  |  |
| --- | --- |
|  |  |
| a) | б) |

Рисунок 2.2-Блок схема функции digitsum(stroka)



Рисунок 2.3 – Итог запуска программы

Листинг 2.1 – Код программы

|  |
| --- |
| #include <iostream>  #include <fstream>  #include "digitsum.h"  #include "filesize.h"  int main()  {  setlocale(LC\_ALL, "ru");  char\* stroka = new char[filesize("stdfile.txt")];  std::cout << "Сумма цифр слов состоящих только из цифр: ";  std::ifstream fin("stdfile.txt");  while (fin.getline(stroka, filesize("stdfile.txt"))) {  digitsum(stroka);  std::cout<<""<<std::endl;  }  fin.close();  } |

Листинг 2.2-библиотека digitsum(stroka)

|  |
| --- |
| #pragma once  #include <fstream>  #include <iostream>  #include "digitsum.h"  void digitsum(char\* stroka)  {  bool flag3 = false;  for (int i = 0; i <= strlen(stroka); i++) {  if (stroka[i] == 32 || (stroka[0] > 47 && stroka[0] < 58 && i==0)) {  int sum = 0;  if (stroka[i] > 47 && stroka[i] < 58 && i==0) {  std::cout << stroka[i];  sum = sum + (int(stroka[0]) - 48);  }  bool flag = true;  while (flag == true) {  for (int k = i + 1; k <= strlen(stroka); k++) {  bool flag1 = false;  if (stroka[k] > 47 && stroka[k] < 58) {  flag1 = true;  }  if (stroka[k] == 0) {  abort;  }  if (flag1 == false) {  flag = false;  i = k - 1;  if (sum != 0) {  flag3 = true; |

Продолжение листинга 2.2

|  |
| --- |
| std::cout <<"=" << sum << " ";  }  break;  }  else {  std::cout << stroka[k];  sum = sum + (int(stroka[k]) - 48);  }  }  }  }  }  if (flag3 == false) {  std::cout << "Таких слов нет.";  }  } |

Задание 3.

В данном задании поставлена задача реализовать соединение символов из первой строки, которые встречаются ровно два раза и второй строки. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках: 3.1, 3.2. По данным блок схемам были написаны программа, которая представлена в листингах 3.1 и 3.2. Результат выполнения программы представлен на рисунке 3.3.

Программа получает первую строку из файла и проверяет на наличие символов, которые встречаются два раза и если они есть, то он получает вторую строку и соединяет символы и строку. Для соединения строк реализована функция lineconnection. После вызова функции на консоль пользователя выводится получившиеся строка.

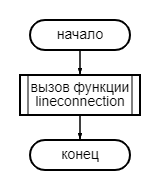


Рисунок 3.1 – Блок схема программы

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рисунок 3.2 – Блок схема функции lineconnection(file\_name)



Рисунок 3.3 – Итог запуска программы

Листинг 3.1 – Код программы 3

|  |
| --- |
| #include <fstream>  #include <iostream>  #include <string>  #include "lineconnection.h"  int main()  {  setlocale(LC\_ALL, "ru");  lineconnection("stdfile.txt");  } |

Листинг 3.2 – библиотека lineconnection(file\_name)

|  |
| --- |
| #pragma once  #include <fstream>  #include <iostream>  #include <string>  void lineconnection(std::string file\_name)  {  std::string stroka;  bool flag = false;  std::string stroki;  std::ifstream fin(file\_name);//открытие файла  if (fin.is\_open()) {//проверка на открытие  std::cout << "Файл открыт успешно." << std::endl;  std::getline(fin, stroka);//получение первой строки  }  else {  std::cout << "Ошибка открытия файла.";  exit(0);  }    for (int i = 0; i < stroka.size(); i++) {//поиск символов которые встречаются два раза  int count = 0;  for (int j = 0; j < stroka.size(); j++) {  if (stroka[i] == stroka[j]) {  count++;  }  }  if (count == 2) {  stroki += stroka[i];  flag = true;  }  }  std::getline(fin, stroka);//получение второй строки  if (flag == true) {//соединение строки |

Продолжение листинга 3.2

|  |
| --- |
| std::cout <<"Полученная строка>>" << stroki + stroka;  }  else {  std::cout << "Такой строки нет.";  }  fin.close();//закрытие файла  } |

Задание 4.

В данном задании поставлена задача реализовать проверку email адреса в файле. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках: 4.1, 4.2. По данным блок схемам были написаны программа, которая представлена в листингах 4.1 и 4.2. Результат выполнения программы представлен на рисунке 4.3.

Программа получает строку в виде email адреса из файла и проверяет его на корректность. Для проверки адреса реализована функция emailcorrection. Также для получения строки из файла используется функция read\_file из первого задания. После вызова функции на консоль пользователя выводится корректность строки или наоборот.

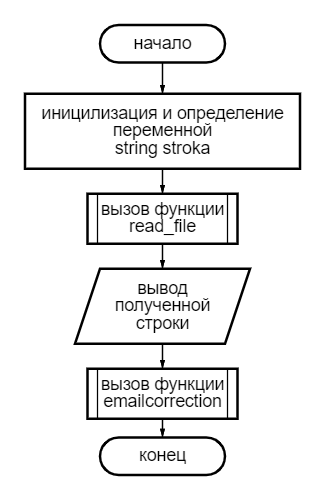


Рисунок 4.1 – Блок схема программы

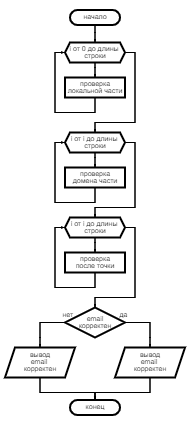


Рисунок 4.2 – Блок схема функции emailcorrection(stroka)



Рисунок 4.3 – Итог запуска программы

Листинг 4.1 – Код программы 4

|  |
| --- |
| #include "readfile.h"  #include <string>  #include <iostream>  #include "emailcorrection.h"  int main()  {  //33 35-39 42-43 45 47 61 63 94-96 123-126 символы 48-57 числа 65-90 97-122 буква  setlocale(LC\_ALL, "ru");  std::string stroka;  stroka=read\_file(stroka,"stdfile.txt");//чтение строки из файла  std::cout << stroka;  emailcorrection(stroka);//проверка Email адреса  } |

Листинг 4.2 – библиотека emailcorrection(stroka)

|  |
| --- |
| #pragma once  #include <string>  #include <iostream>  #include "emailcorrection.h"  void emailcorrection(std::string stroka)  {  //33 35-39 42-43 45 47 61 63 94-96 123-126 символы 48-57 числа 65-90 97-122 буква  int symbol[] = { 33,35,36,37,38,39,42,43,45,47,61,63,94,95,96,123,124,125,126 };//разрешенные символы в ASCII  bool flag1 = false;  bool flag2 = false;  bool flag3 = false;  int sflag = false;  int i = 0;  for (i = 0; i < size(stroka); i++) {//проверка локальной части  sflag = false;  flag1 = false;  for (int j = 0; j < sizeof(symbol); j++) {  if (stroka[i] == symbol[j]) {  flag1 = true;  break;  }  }  if (sflag == false) {  break;  }  if ((stroka[i] < 123 and stroka[i]>96) or (stroka[i] < 58 and stroka[i]>46) or (sflag == true) or (stroka[i] < 91 and stroka[i]>64)) {  flag1 = true;  }  if (stroka[i] == 64) {  flag1 = true;  i++;  break;  }  if (flag1 == false) {  break;  }  }  for (i; i < size(stroka); i++) {//проверка домена  flag2 = false;  if ((stroka[i] < 123 and stroka[i]>96) or (stroka[i] < 58 and stroka[i]>46) or (stroka[i] < 91 and stroka[i]>64) or (stroka[i] == 45) and (sflag==true) and (flag1==true)) {  flag2 = true;  }  if ((stroka[i] == 46) and (size(stroka) - i > 2)) {  flag2 = true;  i++;  break;  }  if (flag2 == false) {  flag2 = false;  break;  }  }  for (i; i < size(stroka); i++) {//проверка после точки  flag3 = false; |

Продолжение листинга 4.2

|  |
| --- |
| if (((stroka[i] < 123 and stroka[i]>96) or (stroka[i] < 58 and stroka[i]>46) or (stroka[i] < 91 and stroka[i]>64)) and (flag2 == true)) {  flag3 = true;  }  }  if (flag3 == true) {//вывод проверки  std::cout << "Email корректен.";  }  else {  std::cout << "Email некорректен.";  }  } |

Задание 5.

В данном задании поставлена задача реализовать генерацию строк длины 10, причем первые 4 символа - цифры, следующие два символы - различные буквы, следующие 4 символа - нули или единицы, причем одна единица точно присутствует. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках: 5.1, 5.2. По данным блок схемам были написаны программа, которая представлена в листингах 5.1 и 5.2. Результат выполнения программы представлен на рисунке 5.3.

Программа генерирует строку по заданному правилу и записывает полученную строку в файл. Для генерации строк реализована функция randomizer. После вызова функции на консоль пользователя выводится сгенерированная строка.

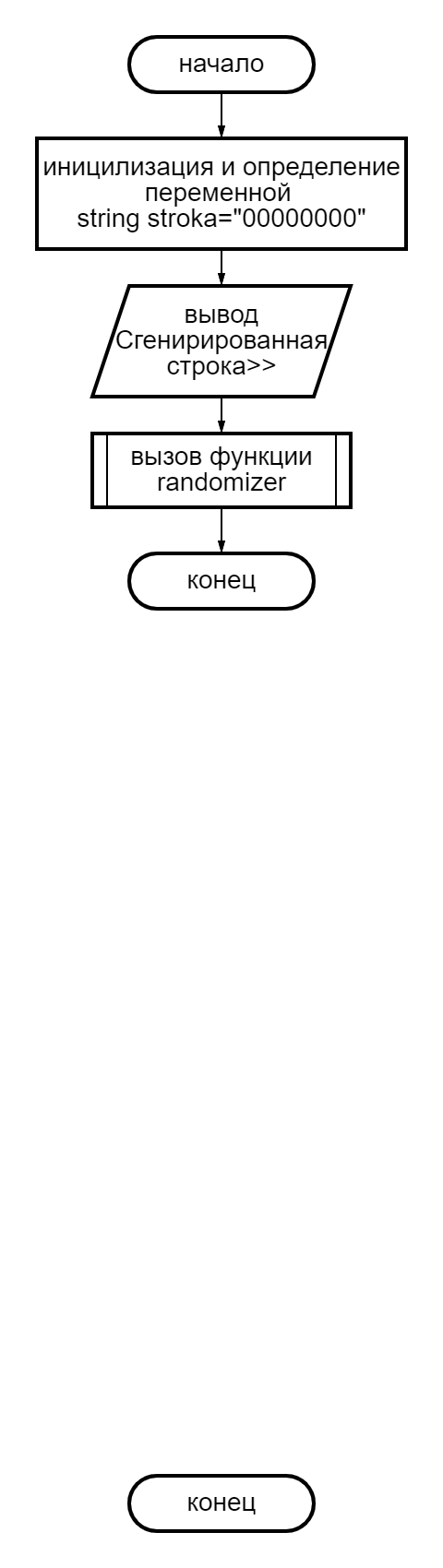


Рисунок 5.1 – Блок схема программы



Рисунок 5.2 – Блок схема функции randomizer(stroka)



Рисунок 5.3 – Итог запуска программы

Листинг 5.1 – Код программы 5

|  |
| --- |
| #include <iostream>  #include <string>  #include "randomizer.h"  int main()  {  setlocale(LC\_ALL, "ru");  std::string stroka{ "0000000000" };  std::cout << "Сгенерированная строка>>";  randomizer(stroka);    } |

Листинг 5.2 – библиотека randomizer(stroka)

|  |
| --- |
| #pragma once  #include <iostream>  #include <string>  #include <random>  void randomizer(std::string stroka)  {  std::random\_device rd;//получение началоьного значения от аппаратного обеспечения  std::mt19937 gen(rd());//создайте объект класса engine  std::uniform\_int\_distribution<> digitdist(48, 57);//диапазон для чисел  std::uniform\_int\_distribution<> symboldist(0, 32);//диапазон для символов  std::uniform\_int\_distribution<> binarydist(48, 49);//диапащон для 0 и 1  int symbol[] = { 33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,58,59,60,61,62,63,64,91,92,93,94,95,96,123,124,125,126 };//массив символов в ASCII  bool flag = false;//флаг для 1  for (int i = 0; i < 4; i++) {//заполение числами  stroka[i] = digitdist(gen);  }  for (int i = 4; i < 6; i++) {//заполение символами  stroka[i] = symbol[symboldist(gen)];  }  while (!flag)  for (int i = 6; i < 10; i++) {//заполение 0 и 1  stroka[i] = binarydist(gen);  if (stroka[i] == 49) {  flag = true;  }  }  std::cout << stroka;  } |

Задание 6.

В данном задании поставлена задача реализовать проверку строки на соответствие расположение букв в алфавитном порядке. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках: 6.1, 6.2. По данным блок схемам были написаны программа, которая представлена в листингах 6.1 и 6.2. Результат выполнения программы представлен на рисунке 6.3.

Программа получает строку из файла и проходится по вей строке для проверки. Для проверки строки реализована функция alphabet. Также для получения строки из файла используется функция read\_file из первого задания. После вызова функции на консоль пользователя выводится корректность строки или наоборот.

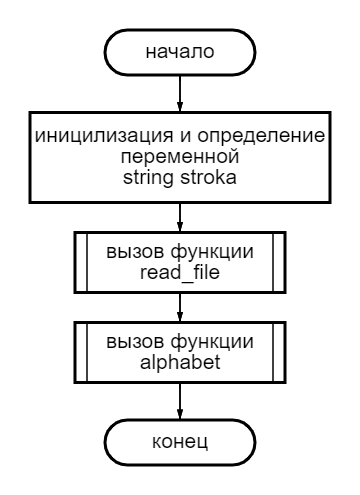


Рисунок 6.1 – Блок схема программы

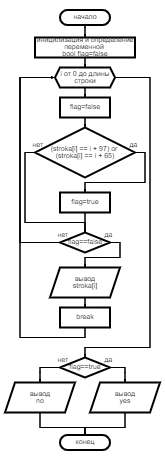


Рисунок 6.2 – Блок схема функции alphabet(stroka)



Рисунок 6.3 – Итог запуска программы

Листинг 6.1 – Код программы 6

|  |
| --- |
| #include "readfile.h"  #include "alphabet.h"  #include <iostream>  #include <string>  int main()  {  setlocale(LC\_ALL, "ru");  std::string stroka;  stroka = read\_file(stroka, "stdfile.txt");  alphabet(stroka);  } |

Листинг 6.2 – библиотека alphabet(stroka)

|  |
| --- |
| #pragma once  #include <string>  #include <iostream>  void alphabet(std::string stroka)  {  bool flag;  for (int i = 0; i < size(stroka); i++) {  flag = false;  if ((stroka[i] == i + 97) or (stroka[i] == i + 65)) {  flag = true;  }  if (flag == false) {  std::cout << stroka[i] << std::endl;  break;  }  }  if (flag) {  std::cout << "yes";  }  else {  std::cout << "no";  }  } |

Задание 7.

В данном задании поставлена задача зашифровывать/дешифровывать строку в файле. Для реализации этой задачи были спроектированы блок схемы, представленные на рисунках: 7.1, 7.2, 7.3. По данным блок схемам были написаны программа, которая представлена в листингах 7.1 и 7.2, 7.3. Результат выполнения программы представлен на рисунке 7.4.

Программа получает строку из файла и по запросу пользователя шифрует или дешифрует строку. Используется алгоритм шифрования Цезаря. Для шифрования/дешифрования реализована функция security, а для записи файла реализована функция rec\_file. Также для получения строки из файла используется функция read\_file из первого задания. После вызова функции полученная строка записывается в файл.

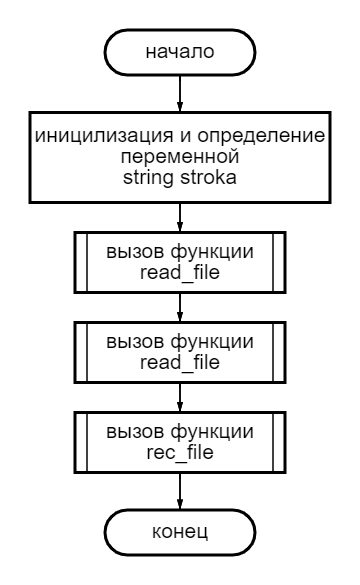


Рисунок 7.1 – Блок схема программы

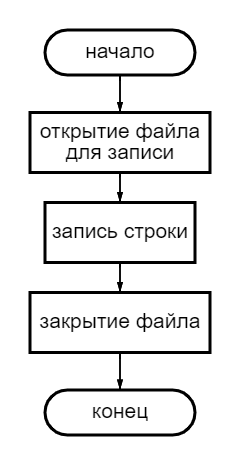


Рисунок 7.2 – Блок схема функции rec\_file(stroka)

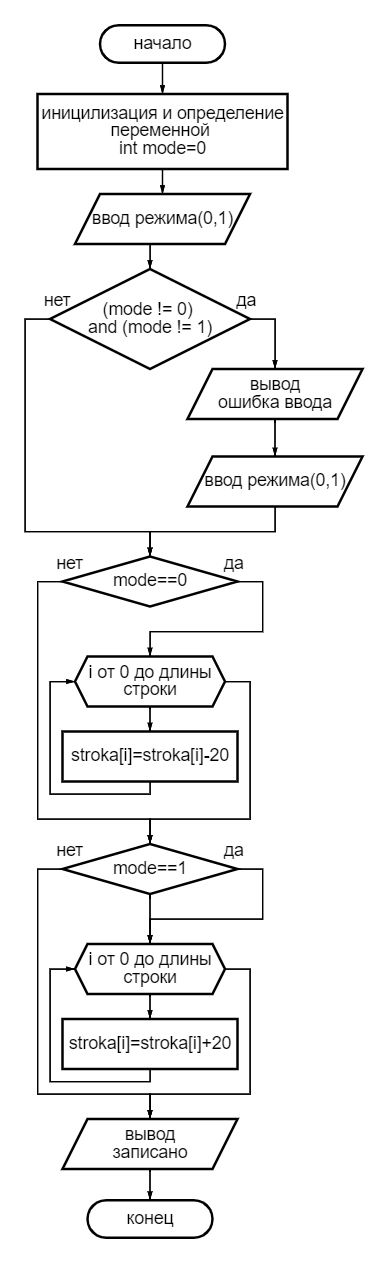


Рисунок 7.3 – Блок схема функции security(stroka)

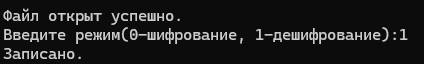


Рисунок 7.4 – Итог запуска программы

Листинг 7.1 – Код программы 7

|  |
| --- |
| #include <iostream>  #include <fstream>  #include "readfile.h"  #include "rec\_file.h"  #include "security.h"  int main()  {  setlocale(LC\_ALL, "ru"); // алгоритм шифрование Цезаря  std::string stroka;  stroka = read\_file(stroka, "stdfile.txt");  stroka = security(stroka);  rec\_file(stroka, "stdfile.txt");  } |

Листинг 7.2 – библиотека rec\_file(stroka)

|  |
| --- |
| #pragma once  #include <iostream>  #include <fstream>  void rec\_file(std::string stroka, std::string file\_name)  {  std::ofstream fout(file\_name);  fout << stroka;  fout.close();  } |

Листинг 7.3 – библиотека security(stroka)

|  |
| --- |
| #pragma once  #include <string>  #include <iostream>  std::string security(std::string stroka)  {  int mode = 0;  std::cout << "Введите режим(0-шифрование, 1-дешифрование):";  std::cin >> mode;  while ((mode != 0) and (mode != 1)) {  std::cout << "Ошибка ввода, только 0 или 1:";  std::cin >> mode;  }  if (mode == 0) {  for (int i = 0; i < size(stroka); i++) {  stroka[i] = stroka[i] - 20;  }  }  if (mode == 1) {  for (int i = 0; i < size(stroka); i++) {  stroka[i] = stroka[i] + 20;  }  }  std::cout << "Записано.";  return stroka;  } |

**Вывод**

В ходе лабораторной работы была изучены информация по работе со строками и файлами. На основе этой информации была проделана работа по написанию кода, представленных на листингах (1.1, 1.2, 1.3, 1.4 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 7.3), для заданий. Также были сделаны блок схемы, которые показаны на рисунках (1.1, 1.2, 1.3, 1.4 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 7.3). В итоге благодаря этой информации можно реализовать различные задания с обработкой строк и файлов.