**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.

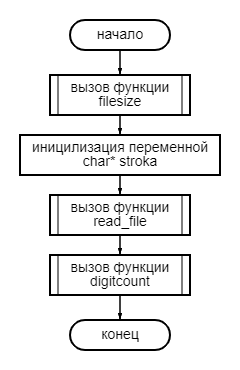


Рисунок 1.1 – Блок схема программы

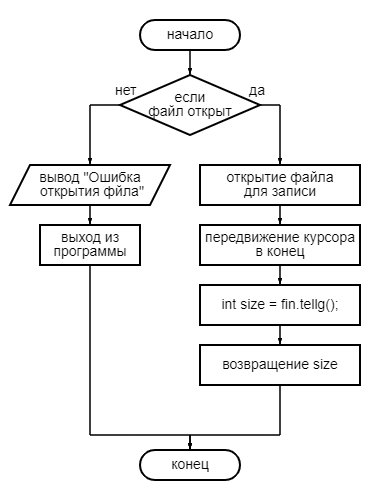


Рисунок 1.2 – Блок схема функции filesize(filename)

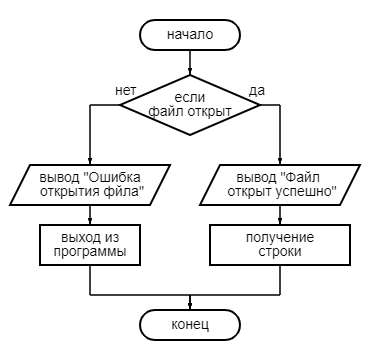


Рисунок 1.3 – Блок схема функции read\_file(filename)

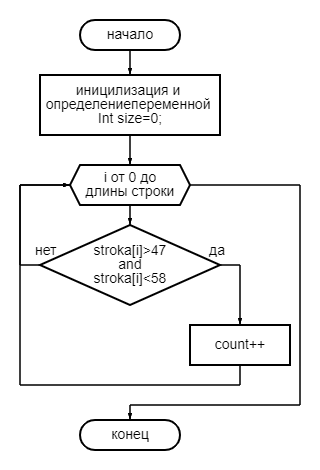


Рисунок 1.4 – Блок схема функции digitcount(stroka)



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 состояшие 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.

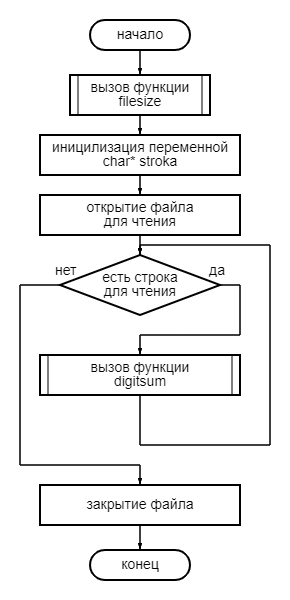


Рисунок 2.1 – Блок схема программы

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

Рисунок 2.2-Блок схема функции digitsum(stroka)



Рисунок 2.2 – 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 замену выбранного пользователем столбца на первый отрицательный столбец. 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 запрашивает размер массива, после которого вводятся элементы массива. Для замены столбцов реализована функция columnreplace. Также для проверки введенных данных использована функция incorrectioninput из первого задания. После вызова функции на консоль пользователя выводится массив с замененными столбцами.

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

Рисунок 3.1 – Блок схема программы

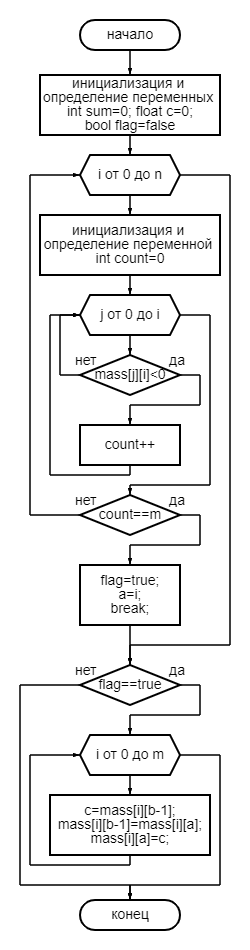


Рисунок 3.2 – Блок схема функции columnreplace(mass, m, n, b)

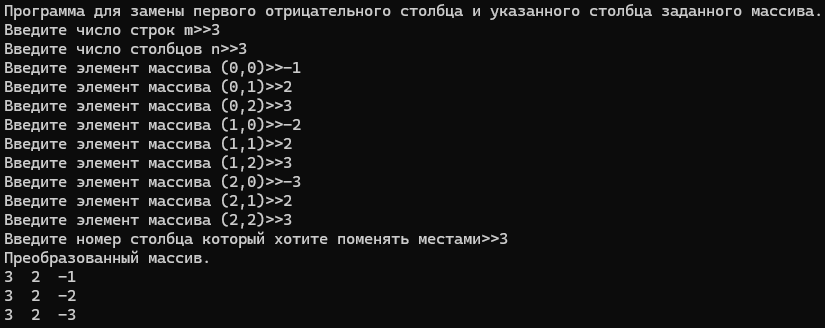


Рисунок 3.4 – Итог запуска программы

Листинг 3.1 – Код программы 3

|  |
| --- |
| #include <iostream>  #include "incorrectioninput.h"  #include "columnreplace.h"  int main()  {  setlocale(LC\_ALL, "ru");  int m = 0, n = 0, a=0, b=0;    std::cout << "Программа для замены первого отрицательного столбца и указанного столбца заданного массива." << std::endl;  std::cout << "Введите число строк m>>";  std::cin >> m;  m = incorrectioninput(m, 1); //проверка m  std::cout << "Введите число столбцов n>>";  std::cin >> n;  n = incorrectioninput(n, 1); //проверка n  float\*\* mass = new float\* [m];  for (int i = 0; i < m; i++) {  mass[i] = new float[n];  }  for (int i = 0; i < m; i++) { //заполнение массива  for (int j = 0; j < n; j++) {  std::cout << "Введите элемент массива " << "(" << i << "," << j << ")>>";  std::cin >> mass[i][j];  mass[i][j] = incorrectioninput(mass[i][j], 2);  }  }  std::cout << "Введите номер столбца который хотите поменять местами>>";//ввод номера столбца  std::cin >> b;  while (b <= 0) {  std::cout << "Номер столбца не может быть отрицательным или равным 0. Введите еще раз>>";  std::cin >> b; b = incorrectioninput(b, 2);  }  while (b > n) {  std::cout << "Номер столбца не может быть больше количества столбцов. Введите еще раз>>";  std::cin >> b; b = incorrectioninput(b, 2);  }  columnreplace(mass, m, n, b); |

Продолжение листинга 3.1

|  |
| --- |
| std::cout << "Преобразованный массив." << std::endl; //вывод преобразованного массива  for (int i = 0; i < m; i++) {  for (int j = 0; j < n; j++) {  std::cout << mass[i][j] << " ";  }  std::cout << '\n';  }  for (int i = 0; i < m; i++) { //очистка массива  delete[] mass[i];  }  delete[] mass;  } |

Листинг 3.2 – библиотека columnreplace

|  |
| --- |
| #pragma once  void columnreplace(float\*\* mass, int m, int n, int b)  {  int a = 0;  bool flag = false;  float c = 0;  for (int i = 0; i < n; i++) { //нахождение первого отрицательного столбца  int count = 0;  for (int j = 0; j < m; j++) {  if (mass[j][i] < 0) {  count++;  }  }  if (count == m) {  flag = true;  a = i;  break;  }  }  if (flag == true) { //замена стобцов  for (int i = 0; i < m; i++) {  c = mass[i][b - 1];  mass[i][b - 1] = mass[i][a];  mass[i][a] = c;  }  }    } |

**Вывод**

В ходе лабораторной работы была изучены информация по работе с двумерными массивами. На основе этой информации была проделана работа по написанию кода, представленных на листингах (1.1, 1.2, 1.3, 2.1, 2.2, 3.1, 3.2), для заданий. Также были сделаны блок схемы, которые показаны на рисунках (1.1, 1.2, 1.3, 2.1, 2.2, 3.1, 3.2). В итоге благодаря этой информации можно реализовать различные задания с обработкой двумерных массивов.