|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Структура данных – строка»** | |
|  | |
| Выполнил студент группы ИКБО-08-21 | Тараканова Е.О. |
| Принял преподаватель | Ермаков С.Р. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# **Цель работы**

Получение навыков в разработке алгоритмов обработки текста – извлечение отдельных элементов, использования средств языка Си и С++ для реализации алгоритмов обработки текстовых данных.

# **Постановка задачи**

1. Разработать программу согласно задаче варианта, используя для представления обрабатываемого в программе текста нуль терминальную строку и средства языка С для выполнения операций над этой строкой.
2. Разработать программу согласно задаче варианта, используя для представления обрабатываемого в программе текста строку string стандартной библиотеки шаблонов и возможности класса для выполнения действий со строкой.
3. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Таблица 1. Упражнения варианта №10.

|  |  |
| --- | --- |
| Упражнение 1 | Дано предложение, состоящее из слов, разделенных запятой или пробелами, среди которых есть группы цифр, определяющих целые числа из диапазона [0..19]. Отредактировать введенную строку, заменив каждое число, на последовательность, заключенную в круглые скобки, символа +, (если число четное) или на – (если число нечетное), длина которой равна найденному числу. |

# **Решение**

**3.1. Null терминальные строки**

Символ – элементарная единица, некоторый набор которых несет определенный смысл. В языке программирования С++ предусмотрено использование символьных констант. Символьная константа – это целочисленное значение (типа int) представленное в виде символа, заключённого в одинарные кавычки, например 'a'. В таблице ASCII представлены символы и их целочисленные значения.

Строки в С++ представляются как массивы элементов типа char, заканчивающиеся нуль-терминатором \0 называются С строками или строками в стиле С.

\0 — символ нуль-терминатора.

Символьные строки состоят из набора символьных констант заключённых в двойные кавычки. При объявлении строкового массива необходимо учитывать наличие в конце строки нуль-терминатора, и отводить дополнительный байт под него.

При объявлении строки не обязательно указывать её размер, но при этом обязательно нужно её инициализировать начальным значением. Тогда размер строки определится автоматически и в конец строки добавится нуль-терминатор.

Строка может содержать символы, цифры и специальные знаки. В С++ строки заключаются в двойные кавычки. Имя строки является константным указателем на первый символ.

**3.2. Cтрока string**

String — класс с методами и переменными для организации работы со строками в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от имени строчного типа данных (англ. string; с англ. — «строка»). В языке C++ и его предшественнике, языке программирования Си, нет встроенной поддержки строкового типа данных, вместо этого используется массив символов. string управляет строками, как и string.h в Си. string использует единственный объект string для организации работы со строками. Являясь частью стандартной библиотеки C++, эти объекты также являются частью стандартного пространства имён — std.

Основные возможности, которыми обладает класс string:

* инициализация массивом символов (строкой встроенного типа) или другим объектом типа string. Встроенный тип не обладает второй возможностью;
* копирование одной строки в другую. Для встроенного типа приходится использовать функцию strcpy();
* доступ к отдельным символам строки для чтения и записи. Во встроенном массиве для этого применяется операция взятия индекса или косвенная адресация с помощью указателя;
* сравнение двух строк на равенство. Для встроенного типа используются функции семейства strcmp();
* конкатенация (сцепление) двух строк, дающая результат либо как третью строку, либо вместо одной из исходных. Для встроенного типа применяется функция strcat(), однако чтобы получить результат в новой строке, необходимо последовательно задействовать функции strcpy() и strcat(), а также позаботиться о выделении памяти;
* встроенные средства определения длины строки (функции-члены класса size() и length()). Узнать длину строки встроенного типа можно только вычислением с помощью функции strlen();
* возможность узнать, пуста ли строка.

**3.3. Реализация упражнения 1 через нуль терминальную строку**

|  |
| --- |
| bool is\_number\_suitable(const char\* num, unsigned short digits)//////  {  if (digits == 0) return false;  char new\_num[1000];  for (int i = 0; i < digits; i++)  new\_num[i] = num[i];  return std::atoi(new\_num) <= 19;  }  bool is\_number\_even(const char\* num, unsigned short digits)//////  {  if (digits == 0) return false;  char new\_num[1000];  for (int i = 0; i < digits; i++)  new\_num[i] = num[i];  int flag = std::atoi(new\_num) % 2;  return (flag == 0); //true = even  }  int the\_number\_itself(const char\* num, unsigned short digits)//////  {  if (digits == 0) return false;  char new\_num[1000];  for (int i = 0; i < digits; i++)  new\_num[i] = num[i];  int flag = std::atoi(new\_num);  return (flag); //true = even  }  // реализуем функцию с учетом того, что на вход мы получаем данные в заведомо верном формате (как указано в условии)  void replace\_numbers(const char\* sentence, char\*\* new\_sentence) ////////  {  const size\_t len = strlen(sentence);  char transformed\_sentence[1000];  unsigned long c = 0;  char num[6];  unsigned short digits = 0;  for (unsigned long i = 0; i < len; i++)  {  if (isdigit(sentence[i]))  {  digits++;  if (digits <= 3)  num[digits - 1] = sentence[i];  if (i + 1 == len)  {  if (is\_number\_suitable(num, digits) && is\_number\_even(num, digits))  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '+';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '+';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '+';  }  }  if (is\_number\_suitable(num, digits) == true && is\_number\_even(num, digits) == false)  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '-';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '-';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '-';  }  }  }  }  else  {  if (is\_number\_suitable(num, digits) && is\_number\_even(num, digits)) // if even  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '+';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '+';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '+';  }  }  else if (is\_number\_suitable(num, digits) == true && is\_number\_even(num, digits) == false)  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '-';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '-';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '-';  }  }  else  {  while (digits)  {  transformed\_sentence[c++] = sentence[i - digits];  digits--;  }  }  transformed\_sentence[c++] = sentence[i];  digits = 0;  }  }  // предложение новой длины  \*new\_sentence = new char[c + 1];  for (unsigned long i = 0; i < c; i++)  (\*new\_sentence)[i] = transformed\_sentence[i];  for (unsigned long i = 0; i < c; i++)  cout << transformed\_sentence[i];  } |

Задача решается при помощи 3 основных функций: is\_number\_suitable проверяет, подходит ли встретившееся в строке число под критерии, указанные в задаче. Если да, возвращает значение true. Её параметрами являются переменные типа const char\* и unsigned short. Далее описываемые функции (за исключением replace\_numbers) для решения этим методом имеют такие же параметры. Функция is\_number\_even возвращает true если встретившееся в строке число чётное. Функция the\_number\_itself возвращает встретившееся в строке число как целочисленное. Все эти функции используются как вложенные в функции replace\_numbers, которая в свою очередь производит непосредственно замену подходящих чётных и нечётных чисел на знаки + и – соответственно. Параметрами для неё являются переменные типа const char\*, char\*\*.

В конце выводится получившаяся строка.

**3.4. Реализация упражнения 1 через строку string**

|  |
| --- |
| string sen(string s) //задание 2  {  bool flag = true;  string tmp = s;  string ans;  regex nums("[^0-9]|^([0-9]+)[^0-9]");  smatch m; //экземпляр шаблона класса match\_results для сопоставления строковых объектов  s += " ";  while (regex\_search(s, m, nums)) //Поиск соответствия регулярному выражению.  {  if (m[1] != "")  {  if (stoi(m[1]) > 19)  {  flag = false;  }  string r = "";  int len = stoi(m[1]);  if (len % 2 == 0)  {  r.insert(0, len, '+');  }  else  {  r.insert(0, len, '-');  }  if (len)  {  ans = tmp.replace(tmp.find(m[1]), m[1].length(), "(" + r + ")");  }  else  {  ans = tmp.replace(tmp.find(m[1]), m[1].length(), "0");  }  }  s = m.suffix();  }  if (flag)  {  cout << ans << "\n";  }  else  {  cout << "Error! Each number in a string should be >= 0 and <= 19!\n";  }  cout << endl;  return ans;  } |

Функция sen производит непосредственно замену подходящих чётных и нечётных чисел на знаки + и – соответственно. Параметром для неё является переменная типа string.

В конце выводится получившаяся строка.

**3.5. Интерфейс**

Изначально пользователю излагается суть задач, которые выполняются данной программой. Далее предоставляется выбор реализации задачи: через нуль терминальную строку или же через строку string. В конце выводится получившаяся строка.

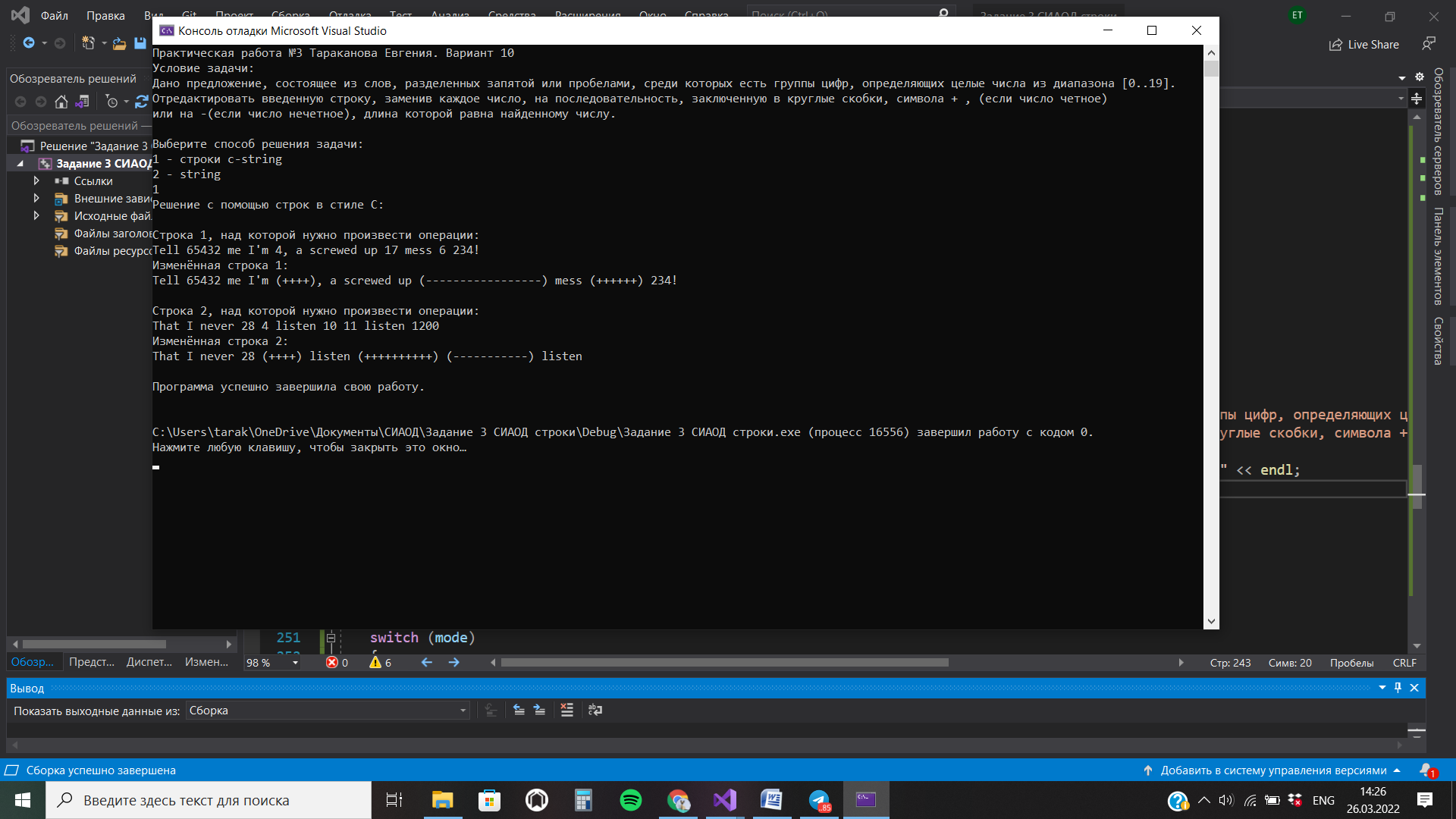


Рисунок 1. Интерфейс

# **Тестирование**

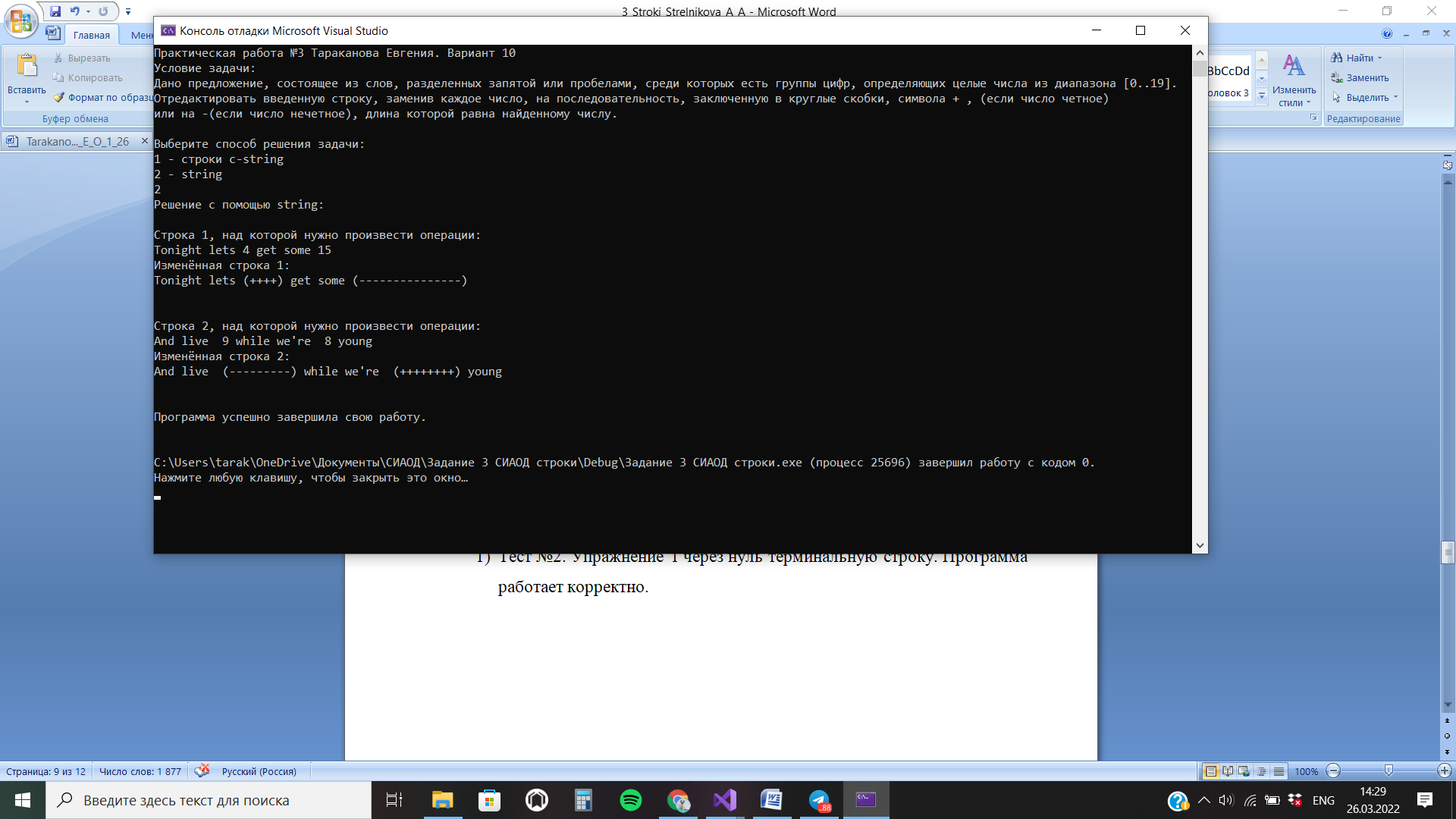
1. Тест №1. Упражнение 2 через строку string. Программа работает корректно. 

Рисунок 2. Тест № 1

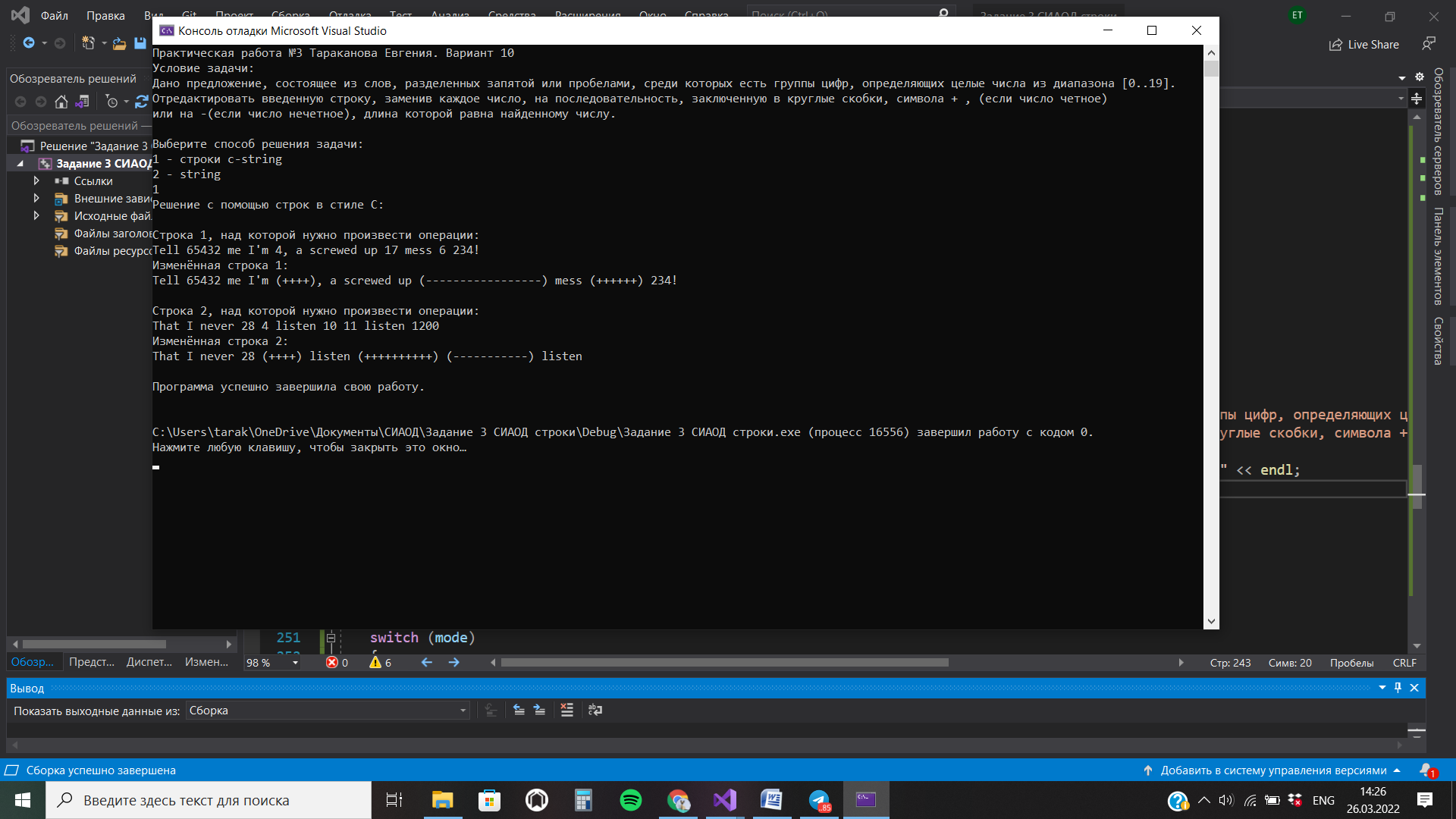
1. Тест №2. Упражнение 1 через нуль терминальную строку. Программа работает корректно. 

Рисунок 3. Тест № 2

# **Вывод**

В результате выполнения работы я получила навыки в разработке алгоритмов обработки текста – извлечение отдельных элементов, использования средств языка Си и С++ для реализации алгоритмов обработки текстовых данных.

# **Исходный код программы**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <string>  #include <regex>  using namespace std;  // 1. Вариант на C-string  // 2. Вариант на std::string  bool is\_number\_suitable(const char\* num, unsigned short digits)//////  {  if (digits == 0) return false;  char new\_num[1000];  for (int i = 0; i < digits; i++)  new\_num[i] = num[i];  return std::atoi(new\_num) <= 19;  }  bool is\_number\_even(const char\* num, unsigned short digits)//////  {  if (digits == 0) return false;  char new\_num[1000];  for (int i = 0; i < digits; i++)  new\_num[i] = num[i];  int flag = std::atoi(new\_num) % 2;  return (flag == 0); //true = even  }  int the\_number\_itself(const char\* num, unsigned short digits)//////  {  if (digits == 0) return false;  char new\_num[1000];  for (int i = 0; i < digits; i++)  new\_num[i] = num[i];  int flag = std::atoi(new\_num);  return (flag); //true = even  }  // реализуем функцию с учетом того, что на вход мы получаем данные в заведомо верном формате (как указано в условии)  void replace\_numbers(const char\* sentence, char\*\* new\_sentence) ////////  {  const size\_t len = strlen(sentence);  char transformed\_sentence[1000];  unsigned long c = 0;  char num[6];  unsigned short digits = 0;  for (unsigned long i = 0; i < len; i++)  {  if (isdigit(sentence[i]))  {  digits++;  if (digits <= 3)  num[digits - 1] = sentence[i];  if (i + 1 == len)  {  if (is\_number\_suitable(num, digits) && is\_number\_even(num, digits))  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '+';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '+';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '+';  }  }  if (is\_number\_suitable(num, digits) == true && is\_number\_even(num, digits) == false)  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '-';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '-';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '-';  }  }  }  }  else  {  if (is\_number\_suitable(num, digits) && is\_number\_even(num, digits)) // if even  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '+';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '+';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '+';  }  }  else if (is\_number\_suitable(num, digits) == true && is\_number\_even(num, digits) == false)  {  int nnuumm = the\_number\_itself(num, digits);  for (int i = 0; i < nnuumm; i++) //вставляем нужное количество символов в зависимости от числа  {  if (i == 0)  {  transformed\_sentence[c++] = '(';  transformed\_sentence[c++] = '-';  }  else if (i == nnuumm - 1)  {  transformed\_sentence[c++] = '-';  transformed\_sentence[c++] = ')';  }  else  transformed\_sentence[c++] = '-';  }  }  else  {  while (digits)  {  transformed\_sentence[c++] = sentence[i - digits];  digits--;  }  }  transformed\_sentence[c++] = sentence[i];  digits = 0;  }  }  // предложение новой длины  \*new\_sentence = new char[c + 1];  for (unsigned long i = 0; i < c; i++)  (\*new\_sentence)[i] = transformed\_sentence[i];  for (unsigned long i = 0; i < c; i++)  cout << transformed\_sentence[i];  }  string sen(string s) //задание 2  {  bool flag = true;  string tmp = s;  string ans;  regex nums("[^0-9]|^([0-9]+)[^0-9]");  smatch m; //экземпляр шаблона класса match\_results для сопоставления строковых объектов  s += " ";  while (regex\_search(s, m, nums)) //Поиск соответствия регулярному выражению.  {  if (m[1] != "")  {  if (stoi(m[1]) > 19)  {  flag = false;  }  string r = "";  int len = stoi(m[1]);  if (len % 2 == 0)  {  r.insert(0, len, '+');  }  else  {  r.insert(0, len, '-');  }  if (len)  {  ans = tmp.replace(tmp.find(m[1]), m[1].length(), "(" + r + ")");  }  else  {  ans = tmp.replace(tmp.find(m[1]), m[1].length(), "0");  }  }  s = m.suffix();  }  if (flag)  {  cout << ans << "\n";  }  else  {  cout << "Error! Each number in a string should be >= 0 and <= 19!\n";  }  cout << endl;  return ans;  }  int main()  {  setlocale(LC\_ALL, "");  cout << "Практическая работа №3 Тараканова Евгения. Вариант 10" << endl;  cout << "Условие задачи: " << endl;  cout << "Дано предложение, состоящее из слов, разделенных запятой или пробелами, среди которых есть группы цифр, определяющих целые числа из диапазона [0..19]." << endl;  cout << "Отредактировать введенную строку, заменив каждое число, на последовательность, заключенную в круглые скобки, символа + , (если число четное) ";  cout << endl << "или на –(если число нечетное), длина которой равна найденному числу." << endl;  cout << endl <<"Выберите способ решения задачи:" << endl << "1 - строки c-string" << endl << "2 - string" << endl;  int mode, size;  do  {  cin >> mode;  if (mode < 1 || mode > 3) cout << "Такого режима нет. Введите число от 1 до 3: ";  } while (mode < 1 || mode > 3);  switch (mode)  {  case 1:  {//строки c-string  cout << "Решение с помощью строк в стиле С: " << endl << endl;  char\* new\_sentence;  cout << "Строка 1, над которой нужно произвести операции: " << endl;  const char\* sentence1 = "Tell 65432 me I'm 4, a screwed up 17 mess 6 234!";  int k = strlen(sentence1);  for (unsigned long i = 0; i < k; i++)  cout << sentence1[i];  cout << endl;  cout << "Изменённая строка 1: " << endl;  replace\_numbers(sentence1, &new\_sentence);  cout << endl << endl;  cout << "Строка 2, над которой нужно произвести операции: " << endl;  const char\* sentence2 = "That I never 28 4 listen 10 11 listen 1200";  int l = strlen(sentence2);  for (unsigned long i = 0; i < l; i++)  cout << sentence2[i];  cout << endl;  cout << "Изменённая строка 2: " << endl;  replace\_numbers(sentence2, &new\_sentence);  cout << endl;  cout << endl << "Программа успешно завершила свою работу." << endl << endl;  break;  }  case 2:  { //string  cout << "Решение с помощью string: " << endl << endl;  string chass1, chass2;  cout << "Строка 1, над которой нужно произвести операции: " << endl;  chass1 = "Tonight lets 4 get some 15";  cout << chass1 << endl;  //getline(cin, chass);  cout << "Изменённая строка 1: " << endl;  sen(chass1);  cout << endl;  cout << "Строка 2, над которой нужно произвести операции: " << endl;  chass2 = "And live 9 while we're 8 young";  cout << chass2 << endl;  cout << "Изменённая строка 2: " << endl;  sen(chass2);  cout << endl;  cout << "Программа успешно завершила свою работу." << endl << endl;  break;  }  }  } |