

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Строки

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-65-23 |  | Петров А. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

# **Оглавление**

[Оглавление 2](#_Toc165481907)

[1. УСЛОВИЕ ЗАДАЧИ. 2](#_Toc165481908)

[2. ЗАДАНИЕ 1. 4](#_Toc165481909)

[2.1. Разработка абстрактного типа данных задачи, используя шаблон 4](#_Toc165481910)

[2.2. Код алгоритма на языке С++. 7](#_Toc165481911)

[2.3. Тестирование алгоритма. 11](#_Toc165481912)

[2.4. Декомпозиция алгоритма для задания 1 12](#_Toc165481913)

[3. ЗАДАНИЕ 2. 12](#_Toc165481914)

[3.1. Код алгоритма на языке С++ 12](#_Toc165481915)

[3.2. Тестирование алгоритма. 15](#_Toc165481916)

[3.3. Декомпозиция алгоритма для задания 1 16](#_Toc165481917)

[4. Выводы 17](#_Toc165481918)

[5. Информационные источники 18](#_Toc165481919)

# УСЛОВИЕ ЗАДАЧИ.

**Задание 1**

Разработать программу, согласно задаче варианта, используя для представления текста нуль терминальную строку (строку, завершающуюся нулем) и средства языка С для выполнения операций над этой строкой. При выполнении операций модификации строки (вставить или удалить подстроку) использовать указатели и функции файла string.h.

**Требования к выполнению задания 1**

1. Изучить функции ввода и вывода текста в строку, завершающуюся

нулем.

2. Ознакомиться с функциями файла string.h, управления строкой,

завершающейся нулем.

3. Требования к разрабатываемой программе.

1) при реализации задачи варианта разработать следующие функции

управления строкой:

− ввод и вывод выполнять функциями для строк, завершающихся нулем;

− выделение слов (часть текста между разделителями) выполнять функцией

strtok (с коррекцией компилятора);

− выделение подстроки в тексте выполнять функциями копирования;

− операции удаления подстроки или вставки подстроки в строку выполнить

применяя указатели и функцию конкатенации;

− массив из слов (подстрок) формировать на основе свободного массива;

2) выполнить декомпозицию задачи варианта (т.е. определить подзадачи,

которые требуется разработать) определить, какие из них могут быть

решены только средствами функций файла string.h, а какие должны быть

реализованы разработанным алгоритмом с применением функций файла

string.h;

3) разработать тесты для тестирования программы в целом;

4) разработать программу, управляющую процессом посредством

текстового меню;

5) выполнить тестирование программы на подготовленных тестах.

Примечание. Под термином слово, в задачах варианта, будем понимать

последовательность символов, заключенную между символами

разделителями.

**Задание 2**

Разработайте программу согласно задаче варианта, используя для представления обрабатываемого в программе текста строку string, стандартной библиотеки шаблонов и возможности класса для выполнения операций со строкой.

**Требования к выполнению задания 2**

1. Изучить функции ввода и вывода текста в строку класса string.

2. Изучить набор перегруженных операторов (знаки операций, которые

можно применять к объектам класса string.

3. Изучить методы класса string, по модификации строки, извлечению

подстрок строки и др.

4. Требования к разрабатываемой программе.

1) ввод и вывод выполнять функциями строк класса string;

2) реализовать задачи, полученные в результате декомпозиции решаемой

задачи в задании 1 данной практической работы, при этом, определить,

какие из них могут быть решены только средствами класса string, а какие

должны быть разработаны вами с применением методов и операторов

класса string; разработать алгоритмы подзадач, требующих разработки,

реализовать с применением методов и операторов класса string;

3) разработать программу, управляющую процессом посредством

текстового меню;

4) выполнить тестирование программы на подготовленных в задании 1

тестах.

**Вариант задачи - № 22**

Даны два предложения. Вывести слова, общие для этих двух предложений. Если

таких слов нет, то вывести сообщение об этом.

# ЗАДАНИЕ 1.

## 2.1. Разработка абстрактного типа данных задачи, используя шаблон

|  |
| --- |
| АТД MyStruct  {  Данные:  Initial\_capacity = 10 – начальная емкость слова  capacity – емкость слова  size – количество слов в строке  strok[1000] - множество символов(входная строка)  \*\*words – свободное множество слов |

Операции:

|  |
| --- |
| **Операция 1**  // Метод, отвечающая за считывание строки с клавиатуры.  //Предусловие: Str& str – входные данные.  //Постусловие: множество str.strok заполненно с клавиатуры.  inputString (Str& str). - заголовок |

|  |
| --- |
| **Операция 2**  // Метод, отвечающая за вывод слов на экран.  //Предусловие: Str& str – входные данные.  //Постусловие: множество str.strok выведено на экран.  print(Str& str). - заголовок |

|  |
| --- |
| **Операция 3**  // Метод, отвечающая за инициализацию всех начальных данных.  //Предусловие: Str& str – входные данные.  //Постусловие: str.capacity = str::intial\_capacity, str.size = 0, выделена память для str.words равная минимальному размеру слова  initStr(Str& str). - заголовок |

|  |
| --- |
| **Операция 4:**  // Метод, отвечающая за очистку памяти.  //Предусловие: Str& str – входные данные.  //Постусловие: память для элементов множества очищена.  freeStr(Str& str). - заголовок |

|  |
| --- |
| **Операция 5:**  // Метод, отвечающая за выделение слов из строки.  //Предусловие: Str& str – входные данные  //Постусловие: свободное множество str.words заполнено словами и переопределена память  selectWords(Str& str). - заголовок |

|  |
| --- |
| **Операция 6:**  //Метод для удаления подстроки из строки  //Предусловие: Str& str; int startIndex, length – входные данные  //Постусловие: Удаление символов в строке начиная с startIndex длинной length  deleteSubstring(Str& str, startIndex, length). - заголовок |

|  |
| --- |
| **Операция 7:**  // Метод для вставки подстроки в строку.  //Предусловие: Str& str, int index, char\* substring – входные данные  //Постусловие: вставляет подстроку в строку в место index  insertSubstring(Str& str, int index, const char\* substring). - заголовок |

|  |
| --- |
| **Операция 8:**  // Метод для выделения подстроки в тексте.  //Предусловие: Str& str; int startIndex, length – входные данные  //Постусловие: возвращает строку  getSubstring(string& str, int startIndex, int length). - заголовок |

|  |
| --- |
| **Операция 9:**  // Метод для вывода слов которые общие для двух строк.  //Предусловие: Str& sentence1, sentence2– входные данные  //Постусловие: выводит слова которые есть в обеих множествах слов из sentence1 и sentence2  findCommonWords(Str& sentence1, Str& sentence2). - заголовок |

|  |
| --- |
| **Операция 10:**  // Метод для вывода массива слов.  //Предусловие: Str& str – входные данные  //Постусловие: выводит слова из массива слов на экран  printWordArray(Str& str). - заголовок |

## 2.2. Код алгоритма на языке С++.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <clocale>  using namespace std;  struct Str {  static const int INITIAL\_CAPACITY = 10;  int capacity;  int size;  char strok[1000];  char\*\* wordArray;  };  // Функции управления строкой  // Метод для ввода строки  void inputString(Str& str);  // Метод для вывода строки  void print(Str& str);  // Метод для вывода массива слов  void printWordArray(Str& str);  // Метод для выделения слов  void selectWords(Str& str);  // Метод для удаления подстроки из строки  void deleteSubstring(Str& str, int startIndex, int length);  // Метод для вставки подстроки в строку  void insertSubstring(Str& str, int index, const char\* substring);  // Метод для выделения подстроки в тексте  char\* getSubstring(const Str& str, int startIndex, int length);  // Метод, отвечающая за инициализацию всех начальных данных.  void initStr(Str& str);  // Метод, отвечающая за очистку памяти.  void freeStr(Str& str);  // меню  void menu(Str& str);  // Метод для вывода слов которые общие для двух предложений  void findCommonWords(Str& sentence1, Str& sentence2);  int main()  {  setlocale(LC\_ALL, "Rus");  Str strok;  inputString(strok);  initStr(strok);  menu(strok);  return 0;  }  void menu(Str& str)  {  char choice;  printf("\nМеню:\n");  printf("1. Ввести строку\n");  printf("2. Вывести строку\n");  printf("3. Получить слова\n");  printf("4. Получить подстроку\n");  printf("5. Удалить подстроку\n");  printf("6. Вставить подстроку\n");  printf("7. Вывести массив слов\n");  printf("8. Общие слова для изначальной строки и произвошльной\n");  printf("0. Выйти\n");  do {  printf("\nВыберите операцию: ");  scanf(" %c", &choice);  getchar();  switch (choice) {  case '1':  inputString(str);  break;  case '2':  print(str);  break;  case '3':  selectWords(str);  break;  case '4': {  int startIndex, length;  printf("Введите начальный индекс и длину подстроки: ");  scanf("%d %d", &startIndex, &length);  getchar();  char\* substring = getSubstring(str, startIndex, length);  if (substring != nullptr) {  printf("Подстрока: %s\n", substring);  free(substring); // освобождаем память, выделенную под подстроку  }  break;  }  case '5': {  int startIndex, length;  printf("Введите начальный индекс и длину подстроки для удаления: ");  scanf("%d %d", &startIndex, &length);  getchar();  deleteSubstring(str, startIndex, length);  printf("Подстрока удалена.\n");  break;  }  case '6': {  int index;  char substring[100];  printf("Введите индекс для вставки подстроки и саму подстроку: ");  scanf("%d %s", &index, substring);  getchar();  insertSubstring(str, index, substring);  printf("Подстрока вставлена.\n");  break;  }  case '7':  printWordArray(str);  break;  case '8':  Str sentence2;  inputString(sentence2);  initStr(sentence2);  findCommonWords(str, sentence2);  break;  case '0':  printf("Программа завершена.\n");  break;  default:  printf("Неверный выбор.\n");  }  } while (choice != '0');  }  void initStr(Str& str)  {  str.capacity = Str::INITIAL\_CAPACITY;  str.size = 0;  str.wordArray = (char\*\*)malloc(str.capacity \* sizeof(char\*));  }  void inputString(Str& str)  {  printf("Запишите вашу строку: ");  fgets(str.strok, sizeof(str.strok), stdin);  for (int i = 0; i < sizeof(str.strok); ++i) {  if (str.strok[i] == '\n') {  str.strok[i] = '\0';  break;  }  }  }  void freeStr(Str& str)  {  for (int i = 0; i < str.size; i++) {  free(str.wordArray[i]);  }  free(str.wordArray);  }  void print(Str& str)  {  printf("Строка: %s\n", str.strok);  }  void selectWords(Str& str)  {  char\* token = strtok(str.strok, " ");  while (token != NULL) {  if (str.size >= str.capacity) {  // Увеличиваем емкость массива wordArray вдвое  str.capacity \*= 2;  str.wordArray = (char\*\*)realloc(str.wordArray, str.capacity \* sizeof(char\*));  }  // Выделение памяти для нового слова  str.wordArray[str.size] = (char\*)malloc((strlen(token) + 1) \* sizeof(char));  // Копирование слова в выделенную память с помощью  strcpy(str.wordArray[str.size], token);  str.size++;  // след слово  token = strtok(NULL, " ");  }  }  void deleteSubstring(Str& str, int startIndex, int length)  {  if (startIndex < 0 || startIndex >= str.size || length <= 0 || startIndex + length > str.size) {  printf("Ошибка: неверные параметры для удаления подстроки.\n");  return;  }  // Сдвигаем символы на место удаленной подстроки  for (int i = startIndex + length; i <= str.size; ++i) {  str.strok[i - length] = str.strok[i];  }  // Обновляем размер строки  str.size -= length;  str.strok[str.size] = '\0'; // Устанавливаем новый нуль  }  void insertSubstring(Str& str, int index, const char\* substring)  {  if (index < 0 || index > str.size) {  printf("Ошибка: неверный индекс для вставки подстроки.\n");  return;  }  int substringLength = strlen(substring);  // Проверяем, достаточно ли памяти для вставки подстроки  if (str.size + substringLength >= sizeof(str.strok)) {  printf("Ошибка: недостаточно места для вставки подстроки.\n");  return;  }  // Сдвигаем символы, чтобы освободить место для вставки  for (int i = str.size; i >= index; --i) {  str.strok[i + substringLength] = str.strok[i];  }  // Копируем подстроку в строку  strcpy(&str.strok[index], substring);  // Обновляем размер строки  str.size += substringLength;  }  char\* getSubstring(const Str& str, int startIndex, int length)  {  if (startIndex < 0 || startIndex >= str.size || length <= 0 || startIndex + length > str.size) {  printf("Ошибка: неверные параметры для выделения подстроки.\n");  return nullptr;  }  // Выделяем память под подстроку  char\* substring = (char\*)malloc((length + 1) \* sizeof(char)); // +1 для нуль-терминатора  if (substring == nullptr) {  printf("Ошибка: не удалось выделить память для подстроки.\n");  return nullptr;  }  // Копируем подстроку из исходной строки  strncpy(substring, &str.strok[startIndex], length);  substring[length] = '\0'; // Устанавливаем нуль-терминатор  return substring;  }  void printWordArray(Str& str)  {  for (int i = 0; i < str.size; ++i) {  printf("\n");  printf("%s", str.wordArray[i]);  printf(" ");  }  }  void findCommonWords(Str& sentence1,Str& sentence2)  {  printf("Общие слова:\n");  bool foundCommon = false;  selectWords(sentence1);  selectWords(sentence2);  // Проходим по всем словам первого предложения  for (int i = 0; i < sentence1.size; ++i) {  // Проходим по всем словам второго предложения  for (int j = 0; j < sentence2.size; ++j) {  // Если находим общее слово, выводим его  if (strcmp(sentence1.wordArray[i], sentence2.wordArray[j]) == 0) {  printf("%s\n", sentence1.wordArray[i]);  foundCommon = true;  break; // переходим к следующему слову первого предложения  }  }  }  if (!foundCommon) {  printf("Общих слов нет.\n");  }  } |

## 2.3. Тестирование алгоритма.

Таблица 1 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Тестирование алгоритма для задания 1 | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | kasdasdqwesd asd  asd | asd |
| 2 | asdda da  asddda daa | Общих слов нет. |

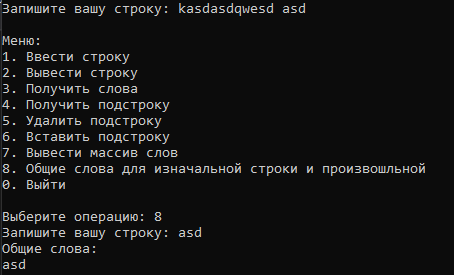


Рисунок 1 - Скриншот тестирования

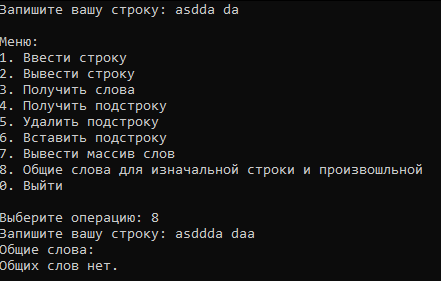


Рисунок 2 - Скриншот тестирования

## 2.4. Декомпозиция алгоритма для задания 1

Нельзя выполнить ни одну из подзадач исключительно с использованием методов из класса string.h, так как даже для ввода значений необходимо заменить обычную строку char на нулевую строку.

# 3. ЗАДАНИЕ 2.

## 3.1. Код алгоритма на языке С++

|  |
| --- |
| #include <iostream>  #include <string>  #include <vector>  using namespace std;  struct Str  {  static const int INITIAL\_CAPACITY = 10;  int size;  string strok;  vector<string> wordArray;  };  // Функции управления строкой  // Метод для ввода строки  void inputString(Str& str);  // Метод для вывода строки  void print(Str& str);  // Метод для вывода массива слов  void printWordArray(Str& str);  // Метод для выделения слов  void selectWords(Str& str);  // Метод для удаления подстроки из строки  void deleteSubstring(string& str, int startIndex, int length);  // Метод для вставки подстроки в строку  void insertSubstring(string& str, int index, string& substring);  // Метод для выделения подстроки в тексте  string getSubstring(string& str, int startIndex, int length);  // меню  void menu(Str& str);  // Метод для вывода слов которые общие для двух предложений  void findCommonWords(Str& sentence1, Str& sentence2);  int main()  {  setlocale(LC\_ALL, "Rus");  Str strok;  inputString(strok);  menu(strok);  return 0;  }  void menu(Str& str)  {  char choice;  int startIndex, length, index;  string substring;  Str sentence2;  cout << "\nМеню:" << endl;  cout << "1. Ввести строку" << endl;  cout << "2. Вывести строку" << endl;  cout << "3. Получить слова" << endl;  cout << "4. Получить подстроку" << endl;  cout << "5. Удалить подстроку" << endl;  cout << "6. Вставить подстроку" << endl;  cout << "7. Вывести массив слов" << endl;  cout << "8. Общие слова для изначальной строки и произвошльной" << endl;  cout << "0. Выйти" << endl;  cout << "\nВыберите операцию: ";  do  {  cin >> choice;  cin.ignore(); // Очищаем буфер ввода от лишних символов  switch (choice) {  case '1':  inputString(str);  break;  case '2':  print(str);  break;  case '3':  selectWords(str);  break;  case '4':  cout << "Введите начальный индекс и длину подстроки: ";  cin >> startIndex >> length;  cout << "Подстрока: " << getSubstring(str.strok, startIndex, length) << endl;  break;  case '5':  cout << "Введите начальный индекс и длину подстроки для удаления: ";  cin >> startIndex >> length;  deleteSubstring(str.strok, startIndex, length);  cout << "Подстрока удалена." << endl;  break;  case '6':  cout << "Введите индекс для вставки подстроки и саму подстроку: ";  cin >> index >> substring;  insertSubstring(str.strok, index, substring);  cout << "Подстрока вставлена." << endl;  break;  case '7':  printWordArray(str);  break;  case '8':  inputString(sentence2);  selectWords(sentence2);  findCommonWords(str, sentence2);  break;  case '0':  cout << "Программа завершена." << endl;  break;  default:  cout << "Неверный выбор." << endl;  }  } while (choice != '0');  }  void inputString(Str& str)  {  cout << "Запишите вашу строку: ";  getline(cin, str.strok);  }  void print(Str& str)  {  cout << "Строка: " << str.strok << endl;  }  void selectWords(Str& str)  {  size\_t pos = 0;  string token;  while ((pos = str.strok.find(' ')) != string::npos) {  token = str.strok.substr(0, pos);  str.wordArray.push\_back(token);  str.strok.erase(0, pos + 1);  }  // Добавим последнее слово в wordArray  str.wordArray.push\_back(str.strok);  }  void printWordArray(Str& str)  {  cout << "Массив слов:" << endl;  for (const auto& word : str.wordArray) {  cout << word << " ";  }  cout << endl;  }  string getSubstring(string& str, int startIndex, int length)  {  if (startIndex < 0 || startIndex >= str.size() || length <= 0 || startIndex + length > str.size()) {  cout << "Ошибка: неверные параметры для выделения подстроки." << endl;  return "";  }  return str.substr(startIndex, length);  }  void deleteSubstring(string& str, int startIndex, int length)  {  if (startIndex < 0 || startIndex >= str.size() || length <= 0 || startIndex + length > str.size()) {  cout << "Ошибка: неверные параметры для удаления подстроки." << endl;  return;  }  str.erase(startIndex, length);  }  void insertSubstring(string& str, int index, string& substring)  {  if (index < 0 || index > str.size()) {  cout << "Ошибка: неверный индекс для вставки подстроки." << endl;  return;  }  str.insert(index, substring);  }  void findCommonWords(Str& sentence1,Str& sentence2)  {  cout << "Общие слова:" << endl;  bool foundCommon = false;  for (const auto& word1 : sentence1.wordArray)  {  for (const auto& word2 : sentence2.wordArray)  {  if (word1 == word2)  {  cout << word1 << endl;  foundCommon = true;  break;  }  }  }  if (!foundCommon)  {  cout << "Общих слов нет." << endl;  }  } |

## 3.2. Тестирование алгоритма.

Таблица 2 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Тестирование алгоритма для задания 2 | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | asd  dsa | Общие слова:  dsa |
| 2 | dsa asd | Общих слов нет. |

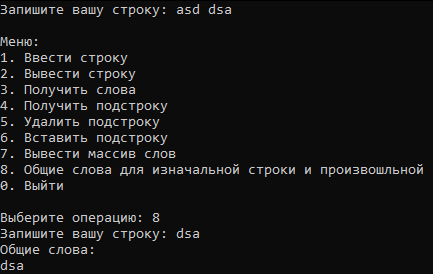


Рисунок 3 - Скриншот тестирования

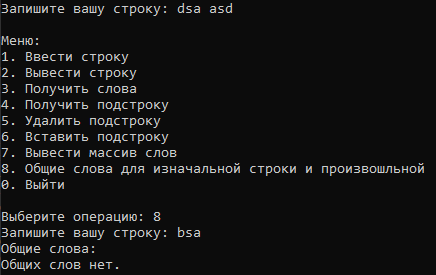


Рисунок 4 - Скриншот тестирования

## 3.3. Декомпозиция алгоритма для задания 1

В этом алгоритме практически все подзадачи могут быть решены с использованием библиотеки "string". Функция ввода может быть реализована с помощью getline(), что устраняет необходимость преобразовывать в нулевую строку. Вывод может быть осуществлен через cout. Однако, разделение на слова не может быть выполнено только с использованием библиотеки string, так как требуется проверка на наличие "слов", которая производится с помощью string::npos. Удаление может быть реализовано с помощью определённого метода.

# 4. Выводы

Во время выполнения задания 1 на пятой практической работе, я изучил библиотеку string.h, работу с динамическими массивами \*\*char и преобразование строк в нулевые строки. Однако, когда я приступил к выполнению задания 2, я освоил современные методы работы со строками, добавленные в стандарт C++, и пришёл к выводу, что использование этих современных методов делает код гораздо более читаемым и компактным.

# 5. Информационные источники

1. Приложение к практическим работам – СДО (online-edu.mirea.ru)

2. Структуры и алгоритмы обработки данных – методические указания / Скворцова Л.А., Филатов А.С., Гусев К.В., Трушин С.М. - Москва, МИРЭА - Российский технологический университет, 2023.