### МИНОБРНАУКИ РОССИИ

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

Кафедра Вычислительной техники

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: «Разработка программы для IBM-совместимой ЭВМ с использованием языка ассемблера и языка высокого уровня»

|  |  |  |
| --- | --- | --- |
| Студент гр. 6891 |  | Гаврилов П.А. |
| Преподаватель |  | Валов А.А. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |
| --- | --- |
| Студент Гаврилов П.А. | |
| Группа 6891 | |
| Тема работы: Разработка программы для IBM-совместимой ЭВМ с использованием языка ассемблера и языка высокого уровня | |
| 1. Техническое задание  Требуется создать на языке высокого уровня функцию MAIN, осуществляющую своевременно вызов программного компонента на языке ассемблера для обработки текстового файла в соответствии с вариантом индивидуального задания. Для обработки текстового файла необходимо содержащийся в нем текст переместить в буфер, созданный в программе. После обработки текста необходимо осуществить вывод отредактированного текста на экран, а затем переместить содержимое буфера в заданный выходной файл.  Исходными данными являются: подлежащий обработке текстовый файл и вводимое с клавиатуры слово (слова). Имя исходного текстового файла задается с клавиатуры. С клавиатуры вводятся также все дополнительные данные, необходимые для решения задачи. Результатом обработки является текст, который выводится на экран и перемещается из буфера в выходной текстовый файл. | |
| 2. Индивидуальные задания | |
| Вариант | Вид обработки |
| 1 | Удалить заданную подстроку во всех строках текста. |

|  |  |  |
| --- | --- | --- |
| Содержание пояснительной записки:  Введение  Функциональная спецификация разрабатываемой программы  Описание алгоритма обработки  Структура программы и обрабатываемых данных  Описание текста программы  Описание пользовательского интерфейса программы  Примеры работы программы  Заключение  Список использованных источников | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Гаврилов П.А. |
| Преподаватель |  | Валов А.А. |

**содержание**

[1. ВВЕДЕНИЕ 5](#_Toc514656164)

[1. ФУНКЦИОНАЛЬНАЯ СПЕЦИФИКАЦИЯ РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ 5](#_Toc514656165)

[2. ОПИСАНИЕ АЛГОРИТМА ОБРАБОТКИ 6](#_Toc514656166)

[3. СТРУКТУРА ПРОГРАММЫ И ОБРАБАТЫВАЕМЫХ ДАННЫХ 7](#_Toc514656167)

[4. ОПИСАНИЕ ТЕКСТА ПРОГРАММЫ 9](#_Toc514656168)

[5. ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ПРОГРАММЫ 14](#_Toc514656169)

[6. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ 15](#_Toc514656170)

[ЗАКЛЮЧЕНИЕ 18](#_Toc514656171)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc514656172)

### ВВЕДЕНИЕ

В данной курсовой работе будет рассмотрено консольное приложение «AWSSEMBLER» для операционной системы Windows, написанное с помощью языков высокого уровня Java, C++ и языка низкого уровня – Macro Assembler (MASM). Задача приложения – обработка строк, считанных из текстового файла.

### 1. ФУНКЦИОНАЛЬНАЯ СПЕЦИФИКАЦИЯ РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ

Программа имеет одну основную функцию – удаление подстроки в строке при ее нахождении в тексте, иначе текст остается без изменений. Путь к исходному файлу, искомая подстрока и путь к файлу, в который будет записан результат, задаются с клавиатуры пользователем.

Программа поддерживает обработку латинских символов, чисел, знаков пунктуации. При работе с текстом чувствительна к регистру.

### 2. ОПИСАНИЕ АЛГОРИТМА ОБРАБОТКИ

(1) Поиск первого вхождения ПОДСТРОКИ в СТРОКУ.

(2) Копирование части СТРОКИ до первого вхождения в БУФЕР.

(3) Обозначение СТРОКОЙ оставшейся части СТРОКИ после вхождения СЛОВА.

(4) Повторение алгоритма с пункта (1), пока найдено вхождение СЛОВА в СТРОКУ.

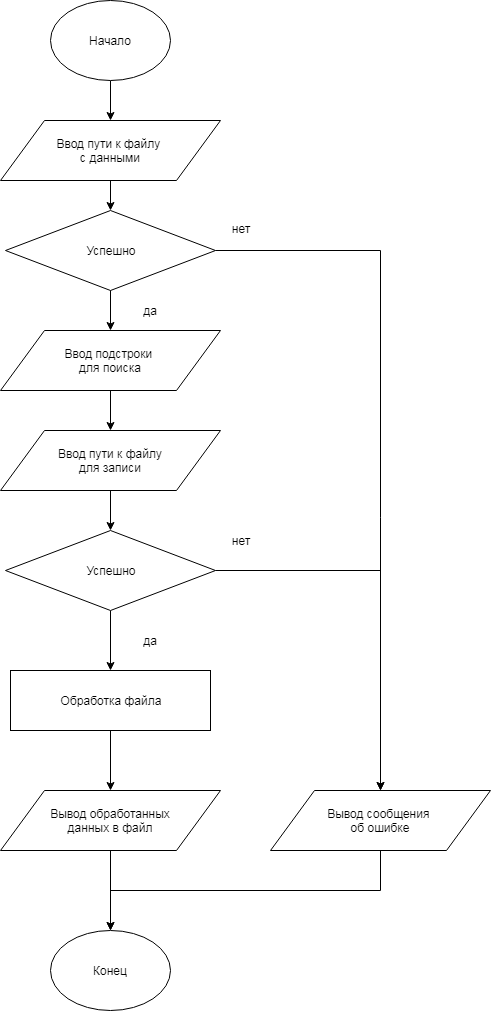
(5) Добавление оставшейся части СТРОКИ в БУФЕР.

Алгоритм обработки реализован с помощью трех функций: функция линейного поиска первого вхождения подстроки в строку «asm\_index\_of», функция определения длинны строки «asm\_string\_length», функция получения части исходной строки по индексам строки «asm\_substring». Эти функции выполнены с использованием вставок на языке Macro Assembler.

### 3. СТРУКТУРА ПРОГРАММЫ И ОБРАБАТЫВАЕМЫХ ДАННЫХ

Приложение «AWSSEMBLER» состоит из:

* Запускаемого приложения на языке Java
* Файлов, содержащих определения и реализации функций на языке C для подключения в java-приложение с помощью интерфейса JNI
* Файлов, содержащих определения и реализации функций непосредственной обработки данных, созданных на языке C с использованием вставок на языке MASM
* Утилиты для сборки дистрибутива приложения

Ниже приведен алгоритм работы программы в целом: 

### 4. ОПИСАНИЕ ТЕКСТА ПРОГРАММЫ

asm\_functions.h

#pragma once  
  
int asm\_string\_length(const char \*);  
  
int asm\_index\_of(const char \*, const char \*);  
  
char \* asm\_subctring(const char \*, int, int);  
  
int asm\_string\_compare(const char \*, const char \*);

asm\_functions.cpp

#include "asm\_functions.h"  
  
/\*\*  
 \* Функция для подсчета символов в строке  
 \*  
 \* text - строка для подсчета символов  
 \*   
 \* возвращает целочисленное значение равное количеству   
 \* символов в строке за исключением символа конца строки  
 \*/  
int asm\_string\_length(const char \* text)  
{  
 int result;  
 \_\_asm {  
 mov edi, text // исходная строка  
 mov eax, 0 // счетчик итераций  
 REPEAT :  
 cmp byte ptr [edi], 0 // сравниваем символ исходной строки с '\0'  
 je COMPLETE // если равны, возвращаем результат  
 inc eax // иначе увеличиваем счетчик  
 inc edi // сдвигаем на следующий символ  
 jmp REPEAT // повторяем  
 COMPLETE :  
 mov result, eax // возвращаем результат  
 }  
 return result;  
  
}  
  
/\*\*  
 \* функция для поиска первого вхождения подстроки в исходной строке  
 \* search\_in - исходная строка в которой ищем  
 \* substring - подстрока, которую ищем в исходной  
 \*  
 \* возвращает целое число, равное индексу первого символа искомого слова в исходной строке  
 \* -1, если подстрока не найдена в исходной сторке  
 \*/   
int asm\_index\_of(const char \* search\_in, const char \* substring)  
{  
 int slen = asm\_string\_length(search\_in);  
 int wlen = asm\_string\_length(substring);  
 int result;  
 \_\_asm {  
 mov esi, search\_in  
 mov edi, substring  
 mov eax, 0h // индекс первого вхождения подстроки  
 mov ecx, slen // длинна исходной сторки  
 COMPARE\_S :  
 mov ebx, ecx // сохранение счетчика итераций по исходной стороке  
 mov edx, esi // сохранение текущей позиции в исходной строке  
 mov ecx, wlen // длина подстроки  
 COMPARE\_W :  
 cmpsb  
 je FOUND  
 jne NOTFOUND  
 FOUND :  
 loop COMPARE\_W  
 jmp COMPARE\_RESULT  
 NOTFOUND :  
 mov esi, edx // восстановить позицию в исходной строке  
 inc esi // переход к следующему символу в исходной сторке  
 mov edi, substring // установить edi на начало подстроки  
 mov ecx, ebx // восстановление счетчика символов в исходной стороке  
 inc eax // увеличиваем счетчик позиции в исходной строке  
 loop COMPARE\_S  
 mov eax, -1 // -1 если подстрока не найдена исходной строке  
 COMPARE\_RESULT:  
 mov result, eax  
 }  
 return result;  
}  
  
/\*\*  
 \* Функция получения подстроки от строки  
 \*   
 \* text - исходный текст  
 \* begin - индекс начала подстроки  
 \* length -длинна подстроки  
 \*  
 \* возвращает новую строку, являющуюся подстрокой исходной строки  
 \*/  
char \* asm\_subctring(const char \* text, int begin, int length)  
{  
 int len = length;  
 char \* result = new char[length + 1];  
 \_\_asm {  
 mov esi, text  
 mov edi, result  
 add esi, begin  
 mov ecx, len  
 REPEAT :  
 mov al, [esi]  
 mov [edi], al  
 inc esi  
 inc edi  
 cmp byte ptr [esi], 0  
 je COMPLETE  
 loop REPEAT  
 COMPLETE :  
 mov [edi], 0  
 }  
 return result;  
}  
  
/\*\*  
 \* Функция сравнения строк  
 \*   
 \* first, second - сравниваемые строки  
 \*  
 \* возвращает целое число: 1 - если строки равны, 0 - если строки не равны  
 \*/  
int asm\_string\_compare(const char \* first, const char \* second)  
{  
 int flen = asm\_string\_length(first);  
 int slen = asm\_string\_length(second);  
 int result;  
 \_\_asm {  
 mov eax, flen  
 mov ebx, slen  
 cmp eax, ebx  
 jne NOT\_EQUAL  
 mov esi, first  
 mov edi, second  
 mov ecx, flen  
 COMPARE :  
 cmpsb  
 jne NOT\_EQUAL  
 inc esi  
 inc edi  
 loop COMPARE  
 mov result, 1  
 jmp COMPLETE  
 NOT\_EQUAL :  
 mov result, 0  
 COMPLETE :  
 }  
 return result;  
}

AsmFunctions.java

package ru.reeson2003.asm;  
  
public class AsmFunctions {  
 public AsmFunctions(String libPath) {  
 System.*loadLibrary*(libPath);  
 }  
  
 public native int asm\_string\_length(String text);  
  
 public native int asm\_index\_of(String text, String substring);  
  
 public native String asm\_substring(String text, int begin, int length);  
}

Program.java

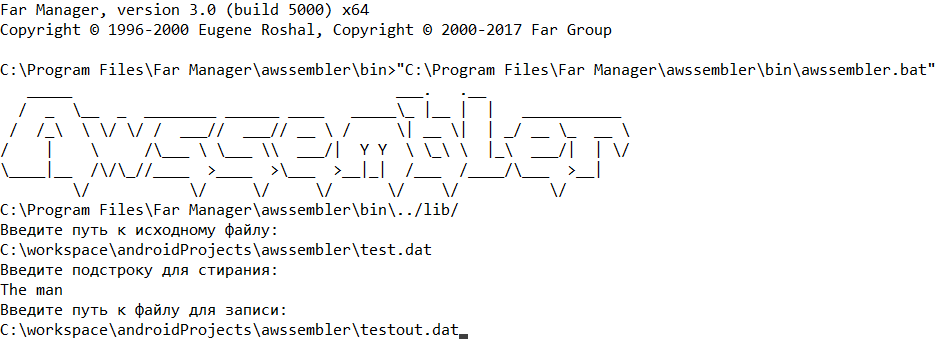
package ru.reeson2003.asm;  
  
import java.io.\*;  
  
*/\*\*  
 \* Date: 27.05.2018.  
 \* Time: 18:51.  
 \*  
 \** ***@author*** *Pavel Gavrilov.  
 \*/*public class Program implements Title{  
 public static void main(String[] args) {  
 try {  
 *printTitle*();  
 SubstringEraser substringEraser = *getSubstringEraser*();  
 BufferedReader isReader = new BufferedReader(new InputStreamReader(System.*in*));  
 System.*out*.println("Введите путь к исходному файлу:");  
 String filePath = isReader.readLine();  
 BufferedReader fileReader = new BufferedReader(new FileReader(filePath));  
 System.*out*.println("Введите подстроку для стирания:");  
 String toErase = isReader.readLine();  
 System.*out*.println("Введите путь к файлу для записи:");  
 filePath = isReader.readLine();  
 FileWriter writer = new FileWriter(filePath);  
 String line;  
 while (true) {  
 line = fileReader.readLine();  
 if (line == null)  
 break;  
 String erased = substringEraser.eraseAll(line, toErase) + '\n';  
 writer.write(erased);  
 System.*out*.println(erased);  
 }  
 writer.close();  
 fileReader.close();  
 isReader.close();  
 } catch (Throwable e) {  
 System.*err*.println("Ошибка!!!");  
 e.printStackTrace();  
 }  
 }  
  
 private static SubstringEraser getSubstringEraser() {  
 System.*out*.println(System.*getProperty*("java.library.path"));  
 AsmFunctions asmFunctions = new AsmFunctions("awssembler");  
 return new AwesomeSubstringEraser(asmFunctions);  
 }  
  
 private static void printTitle() {  
 System.*out*.println(*TITLE*);  
 }  
}

AwesomeSubstringEraser.java

package ru.reeson2003.asm;  
  
*/\*\*  
 \* Date: 27.05.2018.  
 \* Time: 20:24.  
 \*  
 \** ***@author*** *Pavel Gavrilov.  
 \*/*public class AwesomeSubstringEraser implements SubstringEraser {  
 private AsmFunctions asm;  
  
 public AwesomeSubstringEraser(AsmFunctions asm) {  
 this.asm = asm;  
 }  
  
 @Override  
 public String eraseAll(String baseString, String substring) {  
 String toSearch = baseString;  
 int substringLength = asm.asm\_string\_length(substring);  
 StringBuilder stringBuilder = new StringBuilder();  
 while (true) {  
 int i = asm.asm\_index\_of(toSearch, substring);  
 if (i < 0) {  
 stringBuilder.append(toSearch);  
 break;  
 }  
 int toSearchLength = asm.asm\_string\_length(toSearch);  
 String head = asm.asm\_substring(toSearch, 0, i);  
 stringBuilder.append(head);  
 toSearch = asm.asm\_substring(toSearch, i + substringLength, toSearchLength - i - substringLength);  
 }  
 return stringBuilder.toString();  
 }  
}

### 5. ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ПРОГРАММЫ

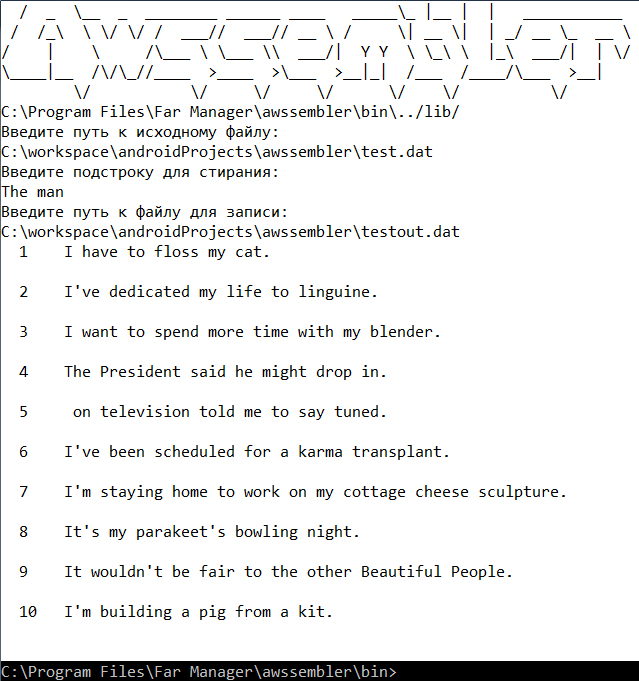
Пользовательский интерфейс представляет из себя окно консоли, в котором последовательно выводятся сообщения:

* Ввести путь к исходному файлу
* Ввести подстроку для поиска
* Ввести путь к файлу для записи

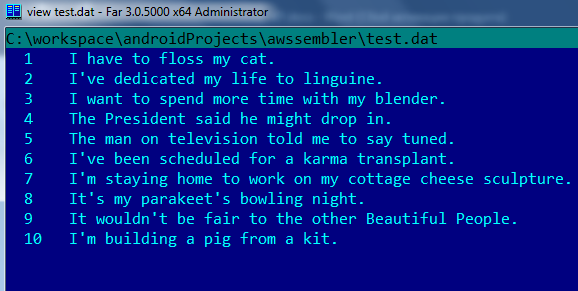
### 6. ПРИМЕР РАБОТЫ ПРОГРАММЫ

Удаление из файла подстроки “The man”

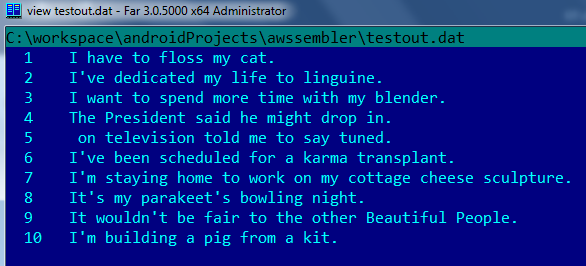
Вывод программы:



Исходный файл:



Полученный файл:



### ЗАКЛЮЧЕНИЕ

В данной курсовой работе было разработано консольное приложение для операционной системы Windows для обработки строк, считанных из текстового файла, написанное с помощью языков высокого уровня Java, C++ и языка низкого уровня – Macro Assembler (MASM).

Приложение было написано и отлажено в средах программирования Intellij Idea, Microsoft Visual Studio 2017. Для сборки приложения используется утилита командной строки Gradle 3.1. По результатам тестирования приложение с возложенными на него задачами справляется.

В курсовой работе приведены алгоритмы работы всей программы в целом и логики работы функции преобразования. Описан интерфейс приложения, приведен пример работы.

При выполнении курсовой работы были приобретены и закреплены навыки использования нескольких языков программирования для создания одного приложения.

Исходный код и инструкции по сборке и запуску приложения доступны в сети по адресу <https://github.com/Reeson2003/awssembler>.

### СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания к проведению лабораторного практикума по дисциплине программирование для подготовки бакалавров по направлению 09.03.01 – «Информатика и вычислительная техника»: СПБГЭТУ «ЛЭТИ», 2016. – 25 с.

2. Основы программирования на языках Си и C++ для начинающих // CppStudio. URL: http://cppstudio.com/cat/309/323/ (дата обращения: 15.05.2018).

3. Собрание материалов по программированию и схемотехнике // Клуб 155. URL: http://www.club155.ru/x86cmdcpu (дата обращения: 15.05.2018).

4. Калашников, О.А. Ассемблер - это просто. Учимся программировать / О.А. Калашников – СПБ: БХВ-Петербург, 2011. – 328 с.