Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Факультет ИВТ

Кафедра вычислительных систем

**Курсовая работа**

на тему "ЧАСТОТНЫЙ АНАЛИЗ ТЕКСТА"

Выполнил:

студент гр. ИС-742

Свириденко В.В.

Проверили:

Старший преподаватель Кафедры ВС

Перышкова Е.Н.

**Новосибирск, 2018**

**Тема курсовой работы**

Частотный анализ текста.

Ссылка на репозиторий ( <https://github.com/HaliksaR/text_analysis_curs> )

**Задание на курсовую работу**

Разработать программу calcFrequency, производящую частотный анализ слов в исходном тексте. На вход программы calcFrequency подается 2 файла. Первый файл содержит текст, который подлежит анализу. Второй файл необходимо создать, и записать все слова, встретившиеся в тексте с указанием частоты появления. Провести сортировку слов по частоте встречаемости (в порядке убывания).

**Вариант усложнения** Добавление графического интерфейса GTK+ 3. Пришлось отказаться от подачи аргументов при запуске программы. Файлы создаются поэтапно, после ввода текса, создается файл с текстом, после следует нажатие кнопки анализ, которая запускает функцию анализа текста из созданного файла с текстом и создает файл с итоговыми значениями анализа. Информация из файла с анализом выводится в заданную область интерфейса. Кнопка очистки стирает данные файлы и области заполнения в интерфейсе.

**Работа алгоритма анализа**

1. Подсчет слов для создания массива структур;
2. Заполнение строки в структурах из потока файла;
3. Корректировка слов (удаление знаков);
4. Подсчет встречаемости каждого слова;
5. Удаление схожих меньших подсчетов и слов, встречающихся меньше двух раз;
6. Сортировка по количеству встречаемости (по убыванию);
7. Вывод в файл.

## Make

make

make open

or add -no-pie

**Требуемые пакеты**

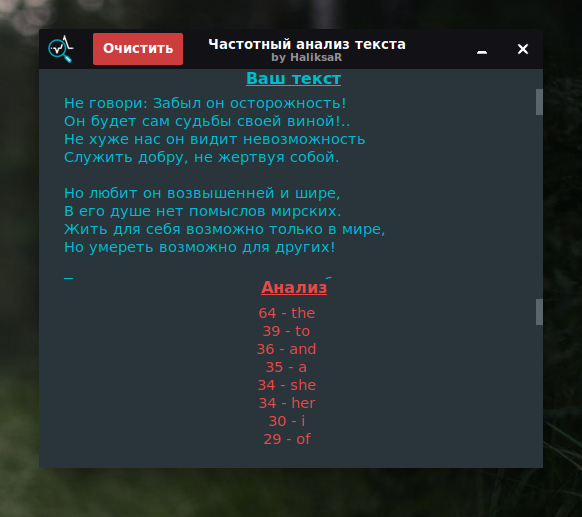
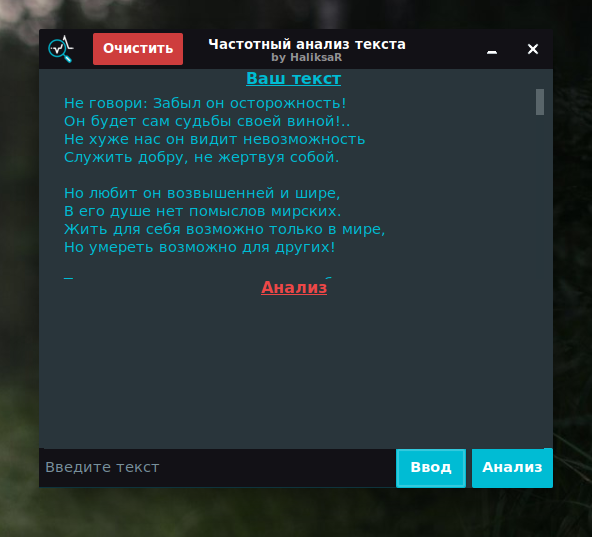
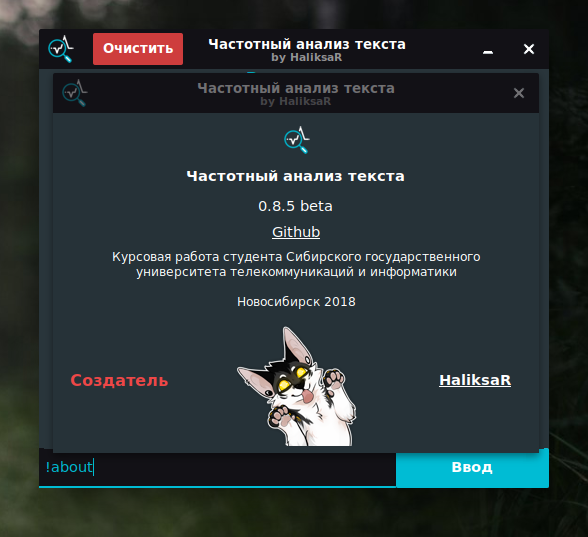
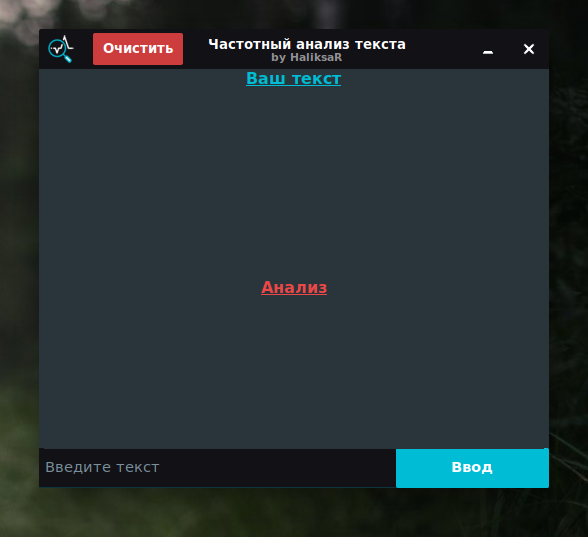
### Ubuntu

apt install libgtk-3-dev

### Solus

eopkg install libgtk-3-devel

**Скриншоты**



**Исходный код**

|  |
| --- |
| **main\_gtk.c** |
| #include "body.h"  #include <gtk/gtk.h>  #include <glib.h>  GtkWidget \*window\_main, \*aboutgtk, \*text\_entry,  \*button\_print, \*button\_analize, \*button\_clear, \*text\_view, \*text\_view2;  GtkBuilder \*builder;  **static** GtkWidget\* **window\_maingtk**();  **static** GtkWidget\* **about**();  **void** **widget\_build**();  **void** **on\_window\_main\_destroy**();  **void** **button\_clear\_clicked**();  **void** **button\_analize\_clicked**();  **void** **button\_print\_clicked**();  **void** **append\_textview**(GtkWidget \*text\_view, **const** gchar \*text);  **void** **error\_analize**();  **int** **main**(**int** argc, **char** \*argv[]) {  gtk\_init(&argc, &argv);  window\_main = window\_maingtk();  gtk\_widget\_show(window\_main);  button\_clear\_clicked();  gtk\_main();  **return** **0**;  }  **static** GtkWidget\* **window\_maingtk**() {  GError\* error = NULL;  builder = gtk\_builder\_new();  **if** (!gtk\_builder\_add\_from\_file(builder, "./style/style.glade", &error)) {  g\_critical("Не могу загрузить файл: %s", error->message);  g\_error\_free(error);  }  GFile \*file = g\_file\_new\_for\_path("./style/style.css");  GtkCssProvider \*provider = gtk\_css\_provider\_new();  **if**(!gtk\_css\_provider\_load\_from\_file(provider, file, &error)) {  g\_warning("%s", error->message);  g\_error\_free(error);  }  gtk\_style\_context\_add\_provider\_for\_screen(gdk\_screen\_get\_default(), GTK\_STYLE\_PROVIDER(provider), GTK\_STYLE\_PROVIDER\_PRIORITY\_USER);  gtk\_builder\_connect\_signals(builder, NULL);  widget\_build();  g\_object\_unref(builder);  **return** window\_main;  }  **static** GtkWidget\* **about**() {  GError\* error = NULL;  builder = gtk\_builder\_new();  **if** (!gtk\_builder\_add\_from\_file(builder, "./style/style.glade", &error)) {  g\_critical("Не могу загрузить файл: %s", error->message);  g\_error\_free(error);  }  GFile \*file = g\_file\_new\_for\_path("./style/style.css");  GtkCssProvider \*provider = gtk\_css\_provider\_new();  **if**(!gtk\_css\_provider\_load\_from\_file(provider, file, &error)) {  g\_warning("%s", error->message);  g\_error\_free(error);  }  gtk\_style\_context\_add\_provider\_for\_screen(gdk\_screen\_get\_default(), GTK\_STYLE\_PROVIDER(provider), GTK\_STYLE\_PROVIDER\_PRIORITY\_USER);  gtk\_builder\_connect\_signals(builder, NULL);  aboutgtk = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "aboutgtk"));  **if** (!aboutgtk) {  g\_critical("Ошибка при получении виджета aboutgtk");  }  g\_object\_unref(builder);  **return** aboutgtk;  }  **void** **widget\_build**() {  window\_main = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "window\_main"));  **if** (!window\_main) {  g\_critical("Ошибка при получении виджета window\_main");  }  text\_entry = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_entry"));  **if** (!text\_entry) {  g\_critical("Ошибка при получении виджета text\_entry");  }  button\_print = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button\_print"));  **if** (!button\_print) {  g\_critical("Ошибка при получении виджета button\_print");  }  button\_analize = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button\_analize"));  **if** (!button\_analize) {  g\_critical("Ошибка при получении виджета button\_analize");  }  button\_clear = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button\_clear"));  **if** (!button\_clear) {  g\_critical("Ошибка при получении виджета button\_clear");  }  text\_view = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_view"));  **if** (!text\_view) {  g\_critical("Ошибка при получении виджета text\_view");  }  text\_view2 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_view2"));  **if** (!text\_view2) {  g\_critical("Ошибка при получении виджета text\_view2");  }  }  **void** **on\_window\_main\_destroy**() {  button\_clear\_clicked();  gtk\_main\_quit();  printf("%s**\n**Goodbye ^^%s**\n**", GREEN, RESET);  }  **void** **button\_clear\_clicked**() {  remove("./src/.data\_text.txt");  remove("./src/.analize\_text.txt");  GtkTextBuffer \*buffer;  buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(text\_view));  gtk\_text\_buffer\_set\_text(buffer,"", **0**);  buffer = gtk\_text\_view\_get\_buffer(GTK\_TEXT\_VIEW(text\_view2));  gtk\_text\_buffer\_set\_text(buffer,"", **0**);  gtk\_widget\_set\_visible(text\_entry, TRUE);  gtk\_widget\_set\_visible(button\_print, TRUE);  gtk\_widget\_set\_visible(button\_analize, FALSE);  gtk\_widget\_set\_size\_request(button\_print, **153**, **40**);  }  **void** **button\_analize\_clicked**() {  data = fopen("./src/.data\_text.txt", "r");  **if** (data != NULL) {  analize = fopen("./src/.analize\_text.txt", "w");  **int** err = analize\_func();  **if** (err == -**1**) {  error\_analize();  } **else** {  fclose(analize);  analize = fopen("./src/.analize\_text.txt", "r");  **char** \*msg = (**char**\*) malloc(**1000**);  **int** num = **0**;  **while** (fscanf(analize, "%s" , msg) != EOF) {  num++;  append\_textview(text\_view2, msg);  append\_textview(text\_view2, " ");  **if** (num == **3**) {  append\_textview(text\_view2, "**\n**");  num = **0**;  }  }  free(msg);  append\_textview(text\_view2, "**\n**");  fclose(analize);  gtk\_widget\_set\_visible(text\_entry, FALSE);  gtk\_widget\_set\_visible(button\_analize, FALSE);  gtk\_widget\_set\_visible(button\_print, FALSE);  fclose(data);  }  } **else** {  error\_analize();  }  }  **void** **button\_print\_clicked**() {  **char** \*text = (**char**\*)gtk\_entry\_get\_text(GTK\_ENTRY(text\_entry));  **if** (strcmp(text, "") != **0**) {  **if**(strcmp(text,"!about") == **0**) {  aboutgtk = about();  gtk\_widget\_show(aboutgtk);  gtk\_entry\_set\_text(GTK\_ENTRY(text\_entry), "");  } **else** {  data = fopen("./src/.data\_text.txt", "a");  append\_textview(text\_view, gtk\_entry\_get\_text(GTK\_ENTRY(text\_entry)));  append\_textview(text\_view, "**\n**");  fprintf (data, "%s**\n**", text);  fclose(data);  gtk\_entry\_set\_text(GTK\_ENTRY(text\_entry), "");  gtk\_widget\_set\_size\_request(button\_print, **70**, **40**);  gtk\_widget\_set\_visible(button\_analize, TRUE);  }  }  }  **void** **append\_textview**(GtkWidget \*text\_view, **const** gchar \*text) {  GtkTextBuffer \*buffer = gtk\_text\_view\_get\_buffer (GTK\_TEXT\_VIEW (text\_view));  GtkTextMark \*mark = gtk\_text\_buffer\_get\_insert (buffer);  GtkTextIter iter;  gtk\_text\_buffer\_get\_iter\_at\_mark (buffer, &iter, mark);  gtk\_text\_buffer\_insert (buffer, &iter, text, -**1**);  }  **void** **error\_analize**() {  append\_textview(text\_view2, "ERROR ANALIZE**\n**");  } |
| **body.h** |
| #ifndef \_\_BODY\_H\_\_  #define \_\_BODY\_H\_\_  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <wchar.h>  #include <wctype.h>  #include <malloc.h>  #define RESET "\033[0m"  #define GREEN "\033[1;32m"  **FILE** \*analize, \*data;  **int** **analize\_func**();  #endif |
| **body.c** |
| #include "body.h"  **typedef** **struct** {  **int** \*same;  **wchar\_t** \*buffer;  } words;  **int** num = **0**;  **int** **scan\_words**(words \*\*arr\_w);  **int** **correct\_words**(words \*\*arr\_w);  **int** **calc\_words**(words \*\*arr\_sravn, words \*\*arr\_correct);  **int** **bun\_words**(words \*\*arr\_sravn, words \*\*arr\_correct, words \*\*arr\_done);  **void** **alignment\_words**(words \*\*arr\_done, **int** max);  **void** **printf\_in\_file**(words \*\*arr\_done, **int** max);  **void** **free\_struct**(words \*\*struct\_v);  words\* **create\_mass**(words \*struct\_v);  words\* **create\_mass**(words \*struct\_v) {  struct\_v = (words\*) malloc(**sizeof**(words));  struct\_v->buffer = (**wchar\_t**\*) malloc(**sizeof**(**wchar\_t**) \* **10000**);  struct\_v->same = (**int**\*) malloc(**sizeof**(**int**));  **if** (struct\_v == NULL || struct\_v->buffer == NULL) {  **return** NULL;  }  **return** struct\_v;  }  **void** **free\_struct**(words \*\*struct\_v) {  **for** (**int** i = **0**; i < num; i++) {  free(struct\_v[i]->same);  free(struct\_v[i]->buffer);  free(struct\_v[i]);  }  }  **int** **scan\_words**(words \*\*arr\_w) {  **int** i = **0**;  **while** (fwscanf(data, L"%ls", arr\_w[i]->buffer) != EOF) {  i++;  }  **if** (correct\_words(arr\_w) == -**1**) {  **return** -**1**;  }  **return** **0**;  }  **int** **correct\_words**(words \*\*arr\_w) {  words \*(arr\_correct[num]);  words \*(arr\_sravn[num]);  **for** (**int** i = **0**; i < num; i++) {  **if** ((arr\_correct[i] = create\_mass(arr\_correct[i])) == NULL) {  **return** -**1**;  }  **if** ((arr\_sravn[i] = create\_mass(arr\_sravn[i])) == NULL) {  **return** -**1**;  }  }  **for** (**int** i = **0**; i < num; i++) {  **for** (**int** j = **0**, k = **0**; j < num; j++) {  **if** (!iswpunct(arr\_w[i]->buffer[j])) {  arr\_correct[i]->buffer[k] = towlower(arr\_w[i]->buffer[j]);  k++;  }  }  wcscpy(arr\_sravn[i]->buffer, arr\_correct[i]->buffer);  }  **if** (calc\_words(arr\_sravn, arr\_correct) == -**1**) {  **return** -**1**;  }  free\_struct(arr\_correct);  free\_struct(arr\_sravn);  **return** **0**;  }  **int** **calc\_words**(words \*\*arr\_sravn, words \*\*arr\_correct) {  words \*(arr\_done[num]);  **for** (**int** i = **0**; i < num; i++) {  **if** ((arr\_done[i] = create\_mass(arr\_done[i])) == NULL) {  **return** -**1**;  }  }  **int** same;  **for** (**int** i = **0**; i < num; i++) {  same = **1**;  **for** (**int** j = **0**; j < num; j++) {  **if** (wcscmp(arr\_sravn[i]->buffer, arr\_correct[j]->buffer) == **0**) {  same++;  }  swprintf(arr\_done[i]->buffer, **10000**, L"%ld - %ls", same, arr\_correct[i]->buffer);  }  }  **if** (bun\_words(arr\_sravn, arr\_correct, arr\_done) == -**1**) {  **return** -**1**;  }  free\_struct(arr\_done);  **return** **0**;  }  **int** **bun\_words**(words \*\*arr\_sravn, words \*\*arr\_correct, words \*\*arr\_done) {  words \*(bun[num]);  **for** (**int** i = **0**; i < num; i++) {  **if** ((bun[i] = create\_mass(bun[i])) == NULL) {  **return** -**1**;  }  }  **int** max = **0**, knok;  **for** (**int** i = **0**; i < num; i++) {  knok = **0**;  **for** (**int** k = **0**; k < num; k++) {  **if** (wcscmp(bun[k]->buffer, arr\_correct[i]->buffer) == **0**) {  knok++;  }  }  arr\_done[max]->same[**0**] = **0**;  **for** (**int** j = **0**; j < num; j++) {  **if** (wcscmp(arr\_sravn[i]->buffer, arr\_correct[j]->buffer) == **0**) {  arr\_done[max]->same[**0**]++;  }  }  **if** (knok == **0** && arr\_done[max]->same[**0**] > **1**) {  swprintf(arr\_done[max]->buffer, **1000**, L"%ld - %ls", arr\_done[max]->same[**0**], arr\_correct[i]->buffer);  wcscpy(bun[max]->buffer, arr\_correct[i]->buffer);  max++;  }  }  free\_struct(bun);  alignment\_words(arr\_done, max);  **return** **0**;  }  **void** **alignment\_words**(words \*\*arr\_done, **int** max) {  **for** (**int** i = **0**; i < max; i++) {  **for** (**int** j = **0**; j < max; j++) {  **if** (arr\_done[i]->same[**0**] > arr\_done[j]->same[**0**]) {  words \*temp = arr\_done[i];  arr\_done[i] = arr\_done[j];  arr\_done[j] = temp;  }  }  }  printf\_in\_file(arr\_done, max);  }  **void** **printf\_in\_file**(words \*\*arr\_done, **int** max) {  **if** (max == **0**) {  fwprintf(analize, L"The same words not found!**\n**");  }  **for** (**int** i = **0**; i < max; i++) {  fwprintf(analize, L"%ls**\n**", arr\_done[i]->buffer);  }  }  **int** **analize\_func**() {  **wchar\_t** \*fal = malloc(**1000**);  **if** (fal == NULL) {  **return** -**1**;  }  **while** (fwscanf(data, L"%ls" , fal) != EOF) {  num++;  }  free(fal);  fclose(data);  words \*(arr\_w[num]);  **for** (**int** i = **0**; i < num; i++) {  **if** ((arr\_w[i] = create\_mass(arr\_w[i])) == NULL) {  **return** -**1**;  }  }  data = fopen("./src/.data\_text.txt", "r");  **if** (data == NULL) {  **return** -**1**;  }  **if** (scan\_words(arr\_w) == -**1**) {  **return** -**1**;  }  free\_struct(arr\_w);  **return** **0**;  } |