Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

Лабораторная работа № 2

«Текстовые файлы»

Проверил: Выполнил:

асс. каф. ЭВС ст. гр. 250504

И.Г. Скиба П.А. Щербо

МИНСК 2023

Цель работы – освоить принципы работы с текстовыми файлами в языке С.

Условие:

Написать программу сжатия текстового файла по алгоритму:

1. Подсчет частоты встречи каждого слова в файле.
2. Поиск самого популярного среди длинных слов (А).
3. Поиск самого непопулярного среди коротких слов (В).
4. Замена всех слов А и В друг на друга.
5. Повтор пунктов 2-4 до тех пор, пока это имеет смысл.

- Программа должна сжать и разжать текстовый файл по приведенному выше алгоритму.

- В результате работы должно быть создано два файла: сжатый и разжатый. Последний должен полностью совпадать с исходным. Сжатый файл должен быть меньше исходного.

- При выборе слова А и В понятия «длинное слово» и «короткое слово» можно трактовать гибко. То есть, «длинное слово» не обязательно «самое длинное слово», главное, чтобы его использование имело положительный эффект.

- Сжатие и разжатие файла реализовать двумя разными программами (два отдельных проекта).

- Следовательно, в сжатый файл должны быть упакованы данные, необходимые для точного разжатия.

- Для реализации пункта 1 алгоритма использовать собственный связный список (стек или очередь).

Код программы:

void compressor(void) {

FILE\* file = NULL;

char\* word = NULL;

char\* new\_word = NULL;

char\* buf = (char\*)calloc(4096, 1);

int index = 0, start = 0;

file = fopen("/Users/pavelshcherbo/Desktop/compress/Zip.txt", "r");

if (file == NULL) {

printf("Error opening file\n");

exit(1);

}

fgets(buf, 4096, file);

while (!feof(file)) {

while ((start = find\_word(buf, &index)) != -1) {

word = take\_word(buf, start);

new\_word = word\_from\_glossary(word);

insert\_from\_glossary(&buf, word, new\_word, &index);

index++;

}

puts\_file(buf);

index = 0;

fgets(buf, 4096, file);

}

fclose(file);

free(buf);

free(word);

free(new\_word);

}

Блок схемы программы (рис.1, рис.2):



Рисунок 1 - Блок схема для выполнения задания №1



Рисунок 2 - Блок схема сжатия

Код сжатия:

int find\_word(char\* string, int\* index) {

while (string[\*index] != '\0') {

if (\*index == 0 && if\_letter(string[\*index])) {

return \*index;

}

else if (string[\*index] == '-' && if\_letter(string[\*index-1]) && if\_letter(string[\*index+1])) {

}

else if (if\_letter(string[\*index]) && !if\_letter(string[\*index-1])) {

return \*index;

}

(\*index)++;

}

return -1;

}

char\* take\_word(char\* str, int start) {

int index = 0;

char\* buf = (char\*)calloc(256, 1);

while (str[start] != '\0') {

if (str[start] == ' ' || str[start] == '\n' || str[start] == ',' || str[start] == ':' || str[start] == ';' || str[start] == '.' || str[start] == '\"' || str[start] == '!' || str[start] == '?' || str[start] == ')') {

buf[index] = '\0';

buf = (char\*)realloc(buf, strlen(buf) + 1);

return buf;

}

buf[index] = str[start];

start++;

index++;

}

buf[index] = '\0';

buf = (char\*)realloc(buf, strlen(buf) + 1);

return buf;

}

char\* check\_stack(stack\* head, char\* word, int size\_of\_word) {

stack\* p = head;

while (p) {

if (!strcmp(word, p->word)) {

p->size += size\_of\_word;

return head;

}

p = p->next;

}

return NULL;

}

void check\_words(stack\*\* head, words\*\* array, int\* size, char\* word, int counter) {

char\* check = NULL;

int size\_of\_word = strlen(word);

if (size\_of\_word < 2) {

return;

}

if (counter > 0) {

size\_of\_word \*= counter;

}

if (counter == 0) {

check = check\_stack(\*head, word, size\_of\_word);

} else {

check = check\_stack(\*head, word, 0);

}

if (check != NULL) {

\*head = check;

return;

}

push\_node(head, word, size\_of\_word + 1);

}

void words\_for\_change(words\*\* array, glossary\*\* arr, int\* size\_of\_words, int\* size\_of\_dictionary) {

glossary word;

int start = ((\*size\_of\_words) - 1);

int end = 0;

while (strlen((\*array)[end].word) <= 4) {

if (strlen((\*array)[start].word) > strlen((\*array)[end].word)) {

if ((\*array)[start].size > (\*array)[end].size) {

word.word\_for\_change = (\*array)[start].word;

word.word\_that\_change = (\*array)[end].word;

push\_dictionary\_in\_array(word, arr, size\_of\_dictionary);

start--;

end++;

}

}

if (strlen((\*array)[start].word) <= strlen((\*array)[end].word)) {

start--;

}

}

glossary\_output(arr, size\_of\_dictionary);

}

void transfer\_words(stack\*\* head, words\*\* array, int\* size\_of\_words) {

words buf;

while ((\*head)) {

buf.word = (\*head)->word;

buf.size = (\*head)->size;

insert\_to\_array(buf, array, size\_of\_words);

(\*head) = (\*head)->next;

}

}

void words\_for\_glossary(stack\*\* head, words\*\* array, glossary\*\* arr, int\* size\_of\_words, int\* size\_of\_dictionary) {

FILE\* file = NULL;

char\* word = NULL;

char\* buf = (char\*)calloc(4096, 1);

int index = 0, start = 0, counter = 0;

file = fopen("/Users/pavelshcherbo/Desktop/compress/Zip.txt", "r");

if (file == NULL) {

printf("Error opening file\n");

exit(1);

}

fgets(buf, 4096, file);

while (!feof(file)) {

while ((start = find\_word(buf, &index)) != -1) {

word = take\_word(buf, start);

counter = count\_word(buf, word, &index);

check\_words(head, array, size\_of\_words, word, counter);

index++;

}

index = 0;

fgets(buf, 4096, file);

}

fclose(file);

free(buf);

free(word);

transfer\_words(head, array, size\_of\_words);

qsort((\*array), (\*size\_of\_words), sizeof(words), compare\_words);

output(array, size\_of\_words);

words\_for\_change(array, arr, size\_of\_words, size\_of\_dictionary);

}

void insert\_glossary(glossary\*\* arr, int\* size\_of\_dictionary) {

FILE\* file = NULL;

file = fopen("/Users/pavelshcherbo/Desktop/compress/compress.txt", "w");

if (file == NULL) {

printf("Error opening file\n");

exit(1);

}

for (int i = 0; i < (\*size\_of\_dictionary); i++) {

fprintf(file, "%s", (\*arr)[i].word\_for\_change);

fprintf(file, "%c", '/');

fprintf(file, "%s\n", (\*arr)[i].word\_that\_change);

}

fprintf(file, "%c\n", '$');

fclose(file);

}

char\* word\_from\_glossary(char\* word) {

FILE\* file = NULL;

char\* buf = (char\*)calloc(1024, 1);

char\* new\_word = NULL;

file = fopen("/Users/pavelshcherbo/Desktop/compress/compress.txt", "r");

if (file == NULL) {

printf("Error opening file\n");

exit(1);

}

fgets(buf, 1024, file);

while (buf[0] != '$') {

if ((new\_word = find\_word\_in\_glossary(buf, word)) != NULL) {

break;

}

fgets(buf, 1024, file);

}

fclose(file);

free(buf);

return new\_word;

}

void insert\_from\_glossary(char\*\* str, char\* word, char\* new\_word, int\* index) {

if (new\_word == NULL) {

return;

}

int difference= strlen(word) - strlen(new\_word), count = 0, start = (\*index), second\_start = 0;

if (difference> 0) {

while (count != strlen(new\_word)) {

(\*str)[start] = new\_word[count];

start++;

count++;

}

second\_start = start;

for (int i = 0; i < difference; i++) {

while ((\*str)[start] != '\0') {

(\*str)[start] = (\*str)[start + 1];

start++;

}

start = second\_start;

}

(\*index) += strlen(new\_word) - 1;

}

else {

difference= strlen(new\_word) - strlen(word);

start = (strlen((\*str)) + 1);

for (int i = 0; i < difference; i++) {

while (start != (\*index)) {

(\*str)[start] = (\*str)[start - 1];

start--;

}

start = strlen((\*str)) + 1;

}

start = (\*index);

while (count != strlen(new\_word)) {

(\*str)[start] = new\_word[count];

start++;

count++;

}

(\*index) += strlen(new\_word) - 1;

}

}

Результат программы (рис.3):

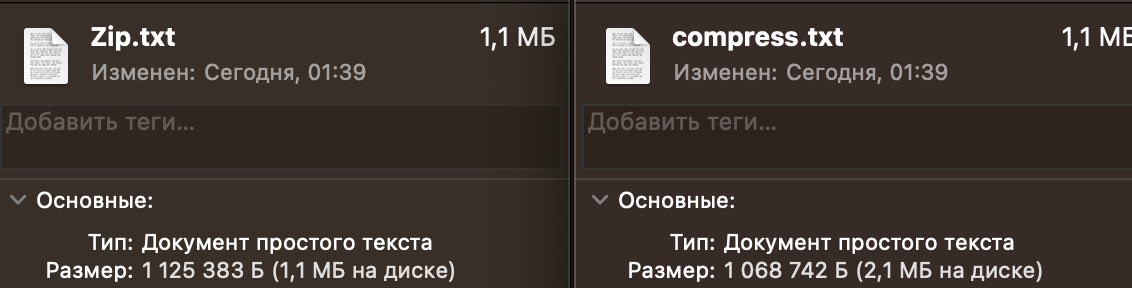


Рисунок 3 - Результат выполнения задания №1

Блок схема (рис.4):



Рисунок 4 - Блок схема для выполнения задания №2

Код программы:

void decompress(words\*\* array, int\* size) {

FILE\* file = NULL;

char\* buf = (char\*)calloc(4096, 1);

char\* word = NULL;

char\* new\_word = NULL;

int index = 0, start = 0;

file = fopen("/Users/pavelshcherbo/Desktop/compress/compress.txt", "r");

if (file == NULL) {

printf("Error opening file\n");

exit(1);

}

fgets(buf, 256, file);

while (buf[0] != '$') {

push\_glossary(array, buf, size);

fgets(buf, 256, file);

}

fgets(buf, 256, file);

while (!feof(file)) {

while ((start = find\_word(buf, &index)) != -1) {

word = take\_word(buf, start);

new\_word = word\_from\_glossary(array, word, size);

insert\_from\_glossary(&buf, word, new\_word, &index);

index++;

}

puts\_file(buf);

index = 0;

fgets(buf, 4096, file);

}

fclose(file);

free(buf);

free(word);

free(new\_word);

}

Блок схема (рис.5):



Рисунок 5 - Блок схема расжатия

Код расжатия:

void words\_output(words\*\* array, const int\* size) {

for (int i = 0; i < (\*size); i++) {

printf("Word1: %s\n", (\*array)[i].change\_word);

printf("Word2: %s\n", (\*array)[i].changed\_word);

}

}

void push\_words\_in\_array(words word, words\*\* array, int\* size) {

(\*size)++;

if ((\*array) == NULL) {

(\*array) = (words\*)malloc((\*size) \* sizeof(word));

}

else {

(\*array) = (words\*)realloc((\*array), (\*size) \* sizeof(word));

}

(\*array)[(\*size) - 1] = word;

}

int find\_word(char\* string, int\* index) {

while (string[\*index] != '\0') {

if (\*index == 0 && if\_letter(string[\*index])) {

return \*index;

}

else if (string[\*index] == '-' && if\_letter(string[\*index-1]) && if\_letter(string[\*index+1])) {

}

else if (if\_letter(string[\*index]) && !if\_letter(string[\*index-1])) {

return \*index;

}

(\*index)++;

}

return -1;

}

char\* take\_word(char\* str, int start) {

int index = 0;

char\* buf = (char\*)calloc(256, 1);

while (str[start] != ' ' && str[start] != '\0' && str[start] != '\n' && str[start] != ',' && str[start] != ':' && str[start] != ';' && str[start] != '.' && str[start] != '\"' && str[start] != '!' && str[start] != '?' && str[start] != ')') {

buf[index] = str[start];

start++;

index++;

}

buf = (char\*)realloc(buf, strlen(buf) + 1);

return buf;

}

char\* word\_from\_glossary(words\*\* array, char\* word, int\* size) {

char\* new\_word = NULL;

for (int i = 0; i < (\*size); i++) {

if (!strcmp(word, (\*array)[i].change\_word)) {

new\_word = (\*array)[i].changed\_word;

break;

}

if (!strcmp(word, (\*array)[i].changed\_word)) {

new\_word = (\*array)[i].change\_word;

break;

}

}

return new\_word;

}

void push\_glossary(words\*\* array, char\* str, int\* size) {

words word;

char\* word1 = (char\*)calloc(256, 1);

char\* word2 = (char\*)calloc(256, 1);

int i = 0, j = 0;

while (str[i] != '/') {

word1[i] = str[i];

i++;

}

i++;

while (str[i] != '\n') {

word2[j] = str[i];

i++;

j++;

}

word1 = (char\*)realloc(word1, (strlen(word1) + 1));

word2 = (char\*)realloc(word2, (strlen(word2) + 1));

word.change\_word = word1;

word.changed\_word = word2;

push\_words\_in\_array(word, array, size);

}

void insert\_from\_glossary(char\*\* str, char\* word, char\* new\_word, int\* index) {

if (new\_word == NULL) {

return;

}

int different = strlen(word) - strlen(new\_word), count = 0, start = (\*index), second\_start = 0;

if (different > 0) {

while (count != strlen(new\_word)) {

(\*str)[start] = new\_word[count];

start++;

count++;

}

second\_start = start;

for (int i = 0; i < different; i++) {

while ((\*str)[start] != '\0') {

(\*str)[start] = (\*str)[start + 1];

start++;

}

start = second\_start;

}

(\*index) += strlen(new\_word) - 1;

}

else {

different = strlen(new\_word) - strlen(word);

start = (strlen((\*str)) + 1);

for (int i = 0; i < different; i++) {

while (start != (\*index)) {

(\*str)[start] = (\*str)[start - 1];

start--;

}

start = strlen((\*str)) + 1;

}

start = (\*index);

while (count != strlen(new\_word)) {

(\*str)[start] = new\_word[count];

start++;

count++;

}

(\*index) += strlen(new\_word) - 1;

}

}

Результат программы (рис.6):

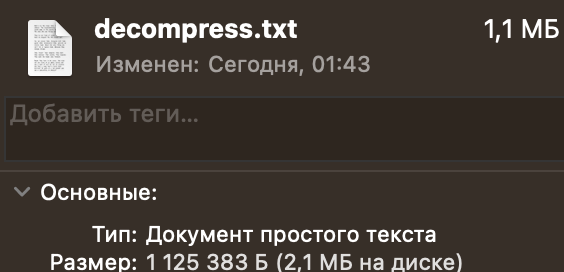


Рисунок 6 - Результат выполнения задания №2