***Кількість слів розділених якимось символом***

int number\_of\_words(string line, char Delimiter) {

int counter = 0; //лічильник кількості слів в рядку

for (int i = 0; i < line.length(); i++) {

if (i != line.length() - 1) //всі, крім останнього символу, тому що рядки не закінчуються на роздільник

{

if (line[i] != Delimiter && line[i + 1] == Delimiter) { //якщо поточний символ не є роздільником, а наступний є

counter++;

}

}

else if(i == line.size() - 1 && line[i]!=Delimiter) counter++; //якщо це останній символ і це не роздільник

}

return counter;

}

***Створення масиву слів***

string\* line\_breakdown(string line, int counter, char Delimiter) {

string\* str\_arr = new string[counter];

int end, //для знаходження наступної потрібної позиції

pos = 0; //лічильник для пошуку позиції символу

for (int number = 0; number < counter; number++) {

while (pos < line.length()) {

if (line[pos] == Delimiter) { //якщо змінна рівна роздільнику то здвигаємо позицію

pos++;

}

else { //інакше шукаємо, скільки символів до наступного роздільника

end = pos;

while (line[end] != Delimiter && end < line.length()) {

end++;

}

str\_arr[number] = line.substr(pos, end-pos);

pos = end;

break;

}

}

}

return str\_arr;

}

***Кількість слів що починаються на певний символ***

int quantity\_of\_words(string line, string symbols) {

int pos = line.find(symbols, 0);

int count = 0;

bool flag = true;;

while (pos != -1) {

if (pos != 0 && line[pos - 1] != ' ') flag = false;

while (pos < line.length()) {

if (line[pos] == ' ' || pos == line.length() - 1) {

if (flag) {

count++;

}

pos = line.find(symbols, pos);

break;

}

else pos++;

}

flag = true;

}

return count;

}

***Вилучення слів довжина яких менша за вказану***

void word\_extraction(string &line, int size) {

int pos = 0,

end = 0;

string word;

while (pos < line.length()) {

if(line[pos]!=' '){

end = pos;

while (end<line.length()&&(line[end] != ' '||end==line.length()-1)) {

end++;

}

word = line.substr(pos, end-pos);

if (word.length() < size) {

line.erase(pos, end-pos);

end-= end - pos;

}

}

else {

pos++;

continue;

}

pos = end;

end = 0;

}

}

***Пошук символу з максимальною частотою***

void maximum\_occurrence(string line) {

char symbol=' ';

int max=0;

int counter;

for (int i = 0; i < line.length(); i++) {

counter = 0;

for (int j = i; j< line.length(); j++) {

if (line[i] == line[j]) counter++;

}

if (counter > max) {

max = counter;

symbol = line[i];

}

}

cout << "Символ с максимальной частотой входимости: "<<symbol << endl;

cout << "Частота входимости:"<<max << endl;

}

***З мінімальною***

void minimum\_occurence(string line) {

char symbol = ' ';

int counter;

int min=line.length();

for (int i = 0; i < line.length(); i++) {

counter = 0;

for (int j = i; j < line.length(); j++) {

if (line[i] == line[j]) counter++;

}

if (min > counter) {

min = counter;

symbol = line[i];

}

}

cout << "Символ с минимальной частотой входимости: " << symbol << endl;

cout << "Частота входимости:" << min << endl;

}

***Підрахунок кількості унікальних символів***

int quantity\_of\_unique\_symbols(string line) {

string unique\_symbols = "";

bool symbol\_unique;

for (int i = 0; i

< line.length(); i++) {

symbol\_unique = true;

for (int j = 0; j < unique\_symbols.length(); j++) {

if (line[i] == unique\_symbols[j]) {

symbol\_unique = false;

break;

}

}

if (symbol\_unique) unique\_symbols += line[i];

}

return unique\_symbols.length();

}

***Кількість кожного символу в строці***

void quantity\_of\_characters(string line) {

string tab = " ";

int quantity=quantity\_of\_unique\_symbols(line);

string\* list = new string[quantity];

int counter;

bool flag=true;

string unique\_symbols = "";

int row = 0;

for (int i = 0; i < line.length(); i++) {

counter = 0;

for (int k = 0; k < unique\_symbols.length(); k++) {

if (line[i] == unique\_symbols[k]) {

flag = false;

break;

}

}

if (flag) {

for (int j = i; j < line.length(); j++) {

if (line[i] == line[j]) counter++;

}

unique\_symbols += line[i];

list[row] = line[i] +tab+ to\_string(counter) +tab+ to\_string((unsigned char)line[i]);

row++;

}

flag = true;

}

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

cout << list[i] << endl;

}

}

***Сортування за алфавітом***

void sort\_array(string\* s, int size) {

int min;

for (int i = 0; i < size - 1; i++) { //проходы алгоритма

for (int j = 0; j < size - i - 1; j++) { //цикл для перебора элементов массива

min = (s[j].size() < s[j + 1].size()) ? s[j].size() : s[j + 1].size();

for (int k = 0; k <= min; k++) {

if (k < min) {

if (tolower(s[j][k]) > tolower(s[j + 1][k])) {

swap\_strings(&s[j], &s[j + 1]);

break;

}

else if (tolower(s[j][k]) < tolower(s[j + 1][k]))

break;

}

else if (k == min && s[j].size() > s[j + 1].size()) //сравниваем кол-во символов в строках

swap\_strings(&s[j], &s[j + 1]);

}

}

}

}

***Строка в форматі аааа->a{4}***

void processing\_list(string\* list, int size) {

int start,

pos,

counter=1;

string push;

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

pos = 0;

while (pos < list[i].length()) {

start = pos;

while (list[i][pos] != ' ' && pos < list[i].length() - 1) {

if (list[i][pos] == list[i][pos + 1]) counter++;

else break;

pos++;

}

if (counter > 1) {

push = "{" + to\_string(counter) + "}";

list[i].replace(start + 1, pos - start, push);

pos = list[i].find\_last\_of("}");

counter = 1;

}

else pos++;

}

}

}