Варыянт 3

Заданне 5



#include <iostream>

#include <fstream>

#include <string>

#include "Windows.h"

void copyAndCountVowels(const std::string& inputFile, const std::string& outputFile, int k) {

std::ifstream file1(inputFile); // Першы файл для чытання, дадзеныя карыстальніка

std::ofstream file2(outputFile); // Другі файл, вывад праграмы

if (!file1.is\_open() || !file2.is\_open()) { // Апрацоўка памылкі адкрыцця файла

std::cerr << "не атрымалася адкрыць " + inputFile + " ці + " + outputFile << std::endl;

return; }

std::string line;

int currentLine = 1; // Лічыльнік для падліку радкоў, каб скапіяваць з k па k + 3

while (std::getline(file1, line)) { // Запісваем у другі файл радкі з k па k + 3 з першага файла

if (currentLine >= k && currentLine <= k + 3) {

file2 << line << std::endl; }

currentLine++;

}

file1.close();

file2.close();

// Адкрываем FILE2 для падліку галосных літар

std::ifstream file2Count(outputFile);

// Апрацоўка памылкі адкрыцця файла

if (!file2Count.is\_open()) {

std::cerr << "Не атрымалася адкрыць файл " + outputFile + " для чытання" << std::endl;

return; }

int vowelCount = 0;

while (file2Count >> line) {

for (char c : line) { // Ітэратар па сімвалу па радку line

if ((('a' <= c && c <= 'z') || ('A' <= c && c <= 'Z')) && // Праверка што гэта літара і што гэта галосная літара (англійскі алфавіт)

(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||

c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')) {

vowelCount++;} } }

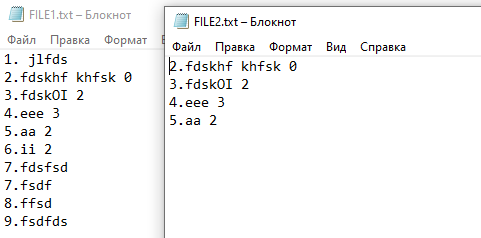
std::cout << "колькасць галосных у FILE2: " << vowelCount << std::endl; }

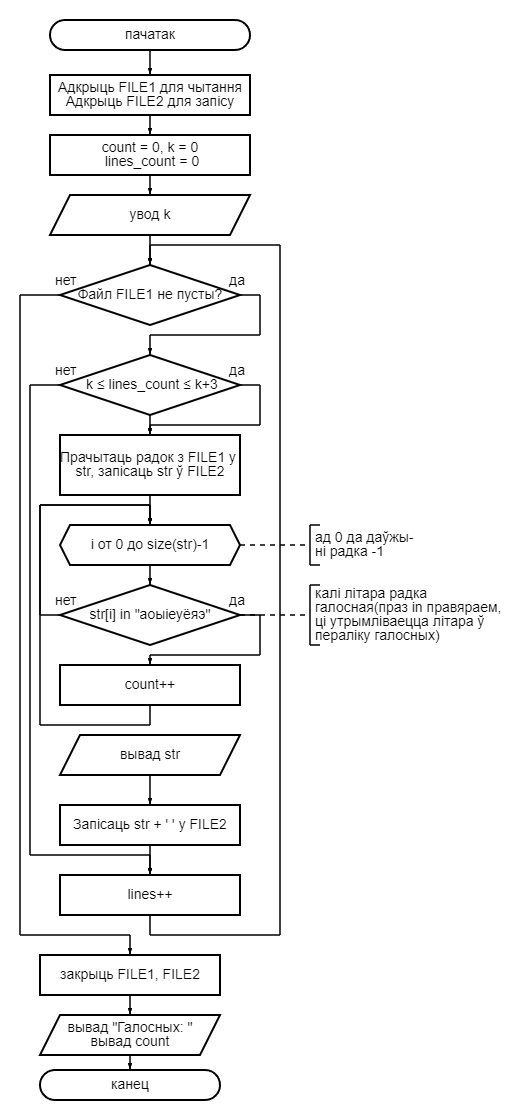
int main() {

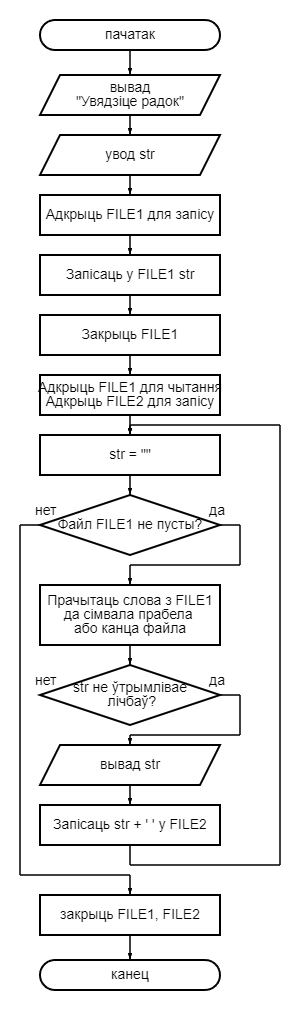
SetConsoleOutputCP(1251);

copyAndCountVowels("FILE1.txt", "FILE2.txt", 2); // для радкоў з 2 па 5 з FILE1

return 0; }







#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include "Windows.h"

using namespace std;

bool cotainsDigits(std::string& str);

void processInputAndWriteToFile(const std::string& fileName, const std::string& input);

std::vector<std::string> readAndProcessFile(const std::string& inputFileName, const std::string& outputFileName);

int main() {

SetConsoleOutputCP(1251);

//Уводзім радок

std::string input;

cout << "Увядзіце радок, які змяшчае словы і лічбы, падзеленыя прабеламі: ";

std::getline(std::cin, input);

processInputAndWriteToFile("inputFile.txt", input);

for (std::string str : readAndProcessFile("inputFile.txt", "outputFile.txt")) { // З дапамогай ітэратара перабіраем ўсе словы ў вектары

cout << str + ' '; // і вводзім гэтыя словы праз прабел

}

return 0;

}

// Запісваем радок у файл

void processInputAndWriteToFile(const std::string& fileName, const std::string& input) {

std::ofstream file(fileName);

if (!file.is\_open()) { // Апрацоўка памылак пры адкрыцці

cerr << "не атрымалася адкрыць файл" << endl;

return;

}

// Запісваем радок у файл

file << input;

file.close();

}

// Чытанне файла і запіс у іншы файл слоў

std::vector<std::string> readAndProcessFile(const std::string& inputFileName, const std::string& outputFileName) {

std::ifstream inputFile(inputFileName);

std::ofstream outputFile(outputFileName);

std::vector<std::string> words; // Вектар для захоўвання слоў

if (!inputFile.is\_open() || !outputFile.is\_open()) {

cerr << "не атрымалася адкрыць файл" << endl;

return words;

}

// Вектар для захоўвання слоў

std::string word;

while (inputFile >> word) {

if (!cotainsDigits(word)) { // Калі няма лічбаў у слове

words.push\_back(word); // Дабаўляем слова ў вектар

outputFile << word << " "; // Захоўваем у файл

}

}

inputFile.close();

outputFile.close();

return words; // Вяртаем вектар з словами

}

// функцыя праверкі ці ёсць лічбы ў слове

bool cotainsDigits(std::string& str) {

for (const char& c : str) { // Канстантны ітэратар па радку str

if ('0' <= c && c <= '9') { // Калі лічба = вяртаем true

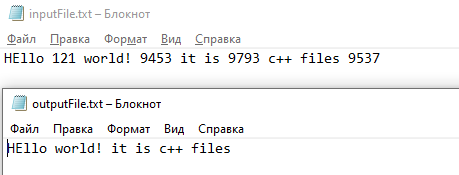
return true;

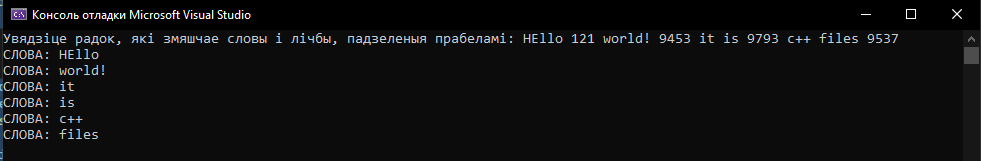
}

}

return false;

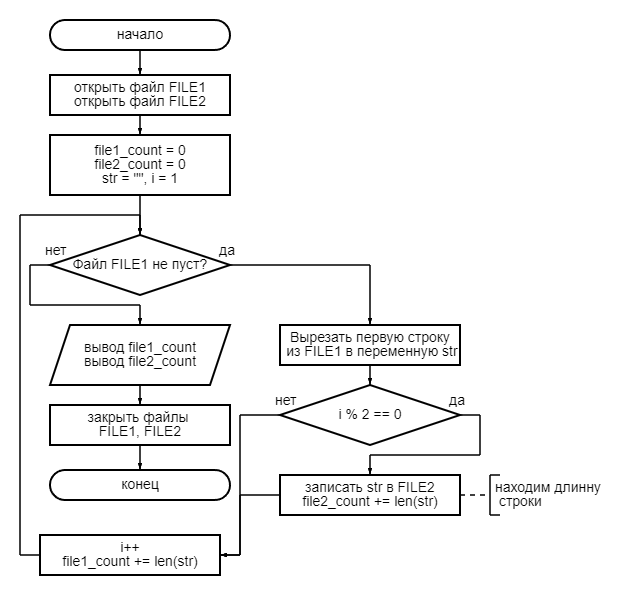
}





Варыянт 1





#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

void copyEvenLines(const std::string& inputFile, const std::string& outputFile);

int countBytes(const std::string& fileName);

int main() {

SetConsoleOutputCP(1251);

std::string file1 = "FILE1.txt";

std::string file2 = "FILE2.txt";

copyEvenLines(file1, file2);

cout << "Памер 1 файла ў байтах: " << countBytes(file1) << endl;

cout << "Памер 2 файла ў байтах: " << countBytes(file2) << endl;

return 0;

}

void copyEvenLines(const std::string& inputFile, const std::string& outputFile) {

std::ifstream in(inputFile);

std::ofstream out(outputFile);

// Праверка на паспяховае адкрыцце файла

if (!in.is\_open() || !out.is\_open()) {

cerr << "Не атрымалася адкрыць файл" << endl;

return;

}

std::string line;

int lineCount = 0;

// Счытваем радкі з файла

while (std::getline(in, line)) {

// Калі нумар радка - цотны лік, то запісваем у іншы файл

if (++lineCount % 2 == 0) {

out << line << std::endl;

}

}

in.close();

out.close();

}

// Функцыя для знаходжання памеру зместу файла ў байтах

int countBytes(const std::string& fileName) {

std::ifstream file(fileName, std::ios::binary); // Адкрываем файл з сцягам std::ios::binary каб escape сімвалы не апрацоўваліся і маглі быць прачытаны

int count = 0; // Лічыльнік

char c; // Бягучы сімвал. не будзе выкарыстоўвацца, але абавязковы для працы get

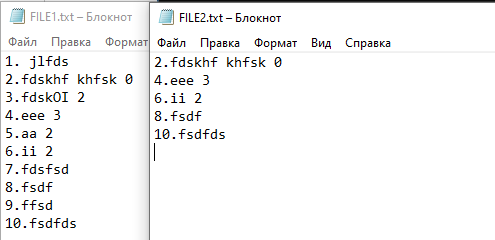
while (file.get(c)) { // Счытваем змест файла пасімвальна

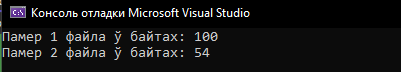
count += sizeof(c); // дадаем да лічыльніка памер у байтах сімвала, атрыманага за дапамогай sizeof

}

return count;

}







#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include "Windows.h"

using namespace std;

bool containsDigits(std::string& str);

void processInputAndWriteToFile(const std::string& fileName, const std::string& input);

std::vector<int> readAndProcessFile(const std::string& inputFileName);

int main() {

SetConsoleOutputCP(1251);

// Уводзім радок

std::string input;

cout << "Увядзіце радок, які змяшчае словы і лічбы, падзеленыя прабеламі: ";

std::getline(std::cin, input);

processInputAndWriteToFile("inputFile.txt", input);

for (int num : readAndProcessFile("inputFile.txt")) { // З дапамогай ітэратара перабіраем усе лікі ў вектары

cout << num << ' '; // і вводзім гэтыя лікі праз прабел

}

return 0;

}

// Запісваем радок у файл

void processInputAndWriteToFile(const std::string& fileName, const std::string& input) {

std::ofstream file(fileName);

if (!file.is\_open()) { // Апрацоўка памылак пры адкрыцці

cerr << "не атрымалася адкрыць файл" << endl;

return;

}

// Запісваем радок у файл

file << input;

file.close();

}

// Чытанне файла і запіс у іншы файл слоў

std::vector<int> readAndProcessFile(const std::string& inputFileName) {

std::ifstream inputFile(inputFileName);

std::vector<int> nums; // Вектар для захоўвання лікаў

if (!inputFile.is\_open()) {

cerr << "не атрымалася адкрыць файл" << endl;

return nums;

}

// Бягучае слова

std::string word;

while (inputFile >> word) {

if (containsDigits(word)) { // у слове не можа быць лікаў. таму калі сустракаюцца лічбы, то гэта лік

int x;

istringstream iss(word); // ствараем аб'ект класа istringstream і ініцыялізуем радком word, які прадстаўляе паток уводу

iss >> x; // канвертуем радок ў лік з дапамогай патоку

if (x % 2 != 0) {

nums.push\_back(x); // Дабаўляем лік у вектар

}

}

}

inputFile.close();

return nums;

}

// функцыя праверкі ці ёсць лічбы ў ліку

bool containsDigits(std::string& str) {

for (const char& c : str) { // Канстантны ітэратар па радку str

if ('0' <= c && c <= '9') { // Калі лічба = вяртаем true

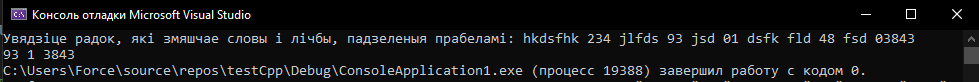
return true;

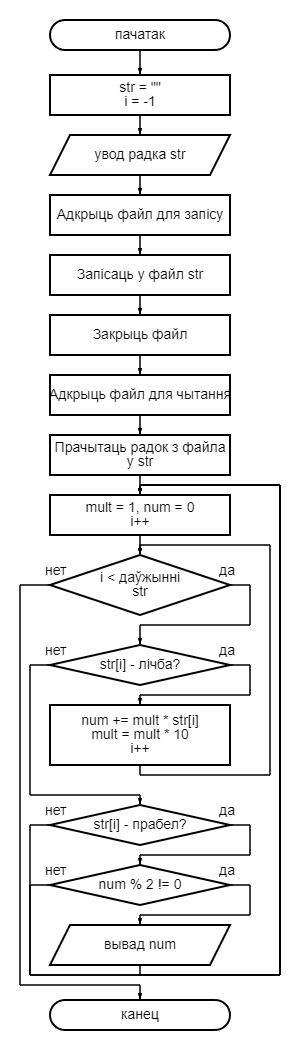
}

}

return false;

}





Варыянт 2



#include <iostream>

#include <fstream>

#include <Windows.h>

int countWordsStartingWithA(const std::string& fileName) {

std::ifstream inputFile(fileName, std::ios::binary); //std::ios::binary у std::ifstream захоўвае дакладнае ўяўленне дадзеных, уключаючы escape-знакі.

if (!inputFile.is\_open()) {

std::cerr << "Не атрымалася адкрыць файл" << std::endl;

return -1;

}

// Вызначэнне памеру файла

// Проста захателася паспрабаваць зрабіць больш цяжкім шляхам

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Функцыя seekg прымае два параметры:

// 1.Позиция, куды трэба ўсталяваць паказальнік.

// 2.Рэжым, які вызначае базавую пазіцыю, адносна якой будзе ўсталяваны паказальнік.

//

// Усталеўваем паказальнік бягучай пазіцыі чытання (get) у патоку ўводу (у нашым выпадку файла) у канец файла выкарыстаючы сцяг std::ios::end

inputFile.seekg(0, std::ios::end);

// Атрымліваем пазіцыю паказальніка, выкарыстоўваючы tellg, які зараз знаходзіцца ў канцы файла. Гэта значэнне адпавядае памеры файла ў байтах. size\_t прызначаны для захоўвання памераў аб'ектаў у байтах ці індэксаў у масівах.

size\_t fileSize = inputFile.tellg();

// Усталеўваем паказальнік у пачатак файла, каб файл можна было прачытаць з пачатку

inputFile.seekg(0, std::ios::beg);

// Вылучаем дынамічную памяць пад масіў сімвалаў памерам fileSize

char\* content = new char[fileSize];

// Счытваем сімвалы з файла ў дынамічны масіў content з выкарыстаннем метаду read

inputFile.read(content, fileSize);

// Закрываем файл

inputFile.close();

// Адкрываем файл для запісу

std::ofstream outputFile("FILE2.txt");

if (!outputFile.is\_open()) {

std::cerr << "Не атрымалася адкрыць файл для запісу" << std::endl;

delete[] content; // Вызваляем памяць перад выхадам з праграмы

return -1;

}

// Капіраванне радкоў, якія пачынаюцца з літары "А" у FILE2

size\_t i = 0;

while (i < fileSize) {

// Знаходзім пачатак радка

if (content[i] == '\n') {

i++;

continue;

}

// Правяраем, ці пачынаецца радок з літары "А"

if (i < fileSize && (content[i] == 'A')) {

// Капіюем радок усярэдзіне гэтага цыклу

while (i < fileSize && content[i] != '\n') {

outputFile << content[i];

++i;

}

}

else {

// Прапускаем радок, які не пачынаецца з літары "А" шляхам "перамотванні" радка (індэкса масіва) да наступнага знака новага радка(\n), што і будзе азначаць канец гэтага радка

while (i < fileSize && content[i] != '\n') {

++i;

}

}

}

outputFile.close();

// Вызваленне памяці

delete[] content;

std::ifstream outputFile2("FILE2.txt");

if (!outputFile2.is\_open()) {

std::cerr << "Не атрымалася адкрыць FILE2.txt" << std::endl;

return 1;

}

int wordCount = 0;

std::string word;

// Падлік колькасці слоў у FILE2 шляхам счытвання слоў, падзеленых прабелам ці \n у радок word

while (outputFile2 >> word) {

++wordCount; // Пры кожным счытванні павялічваем лічыльнік

}

outputFile2.close();

return wordCount;

}

int main() {

SetConsoleOutputCP(1251);

int wordCount = countWordsStartingWithA("FILE1.txt");

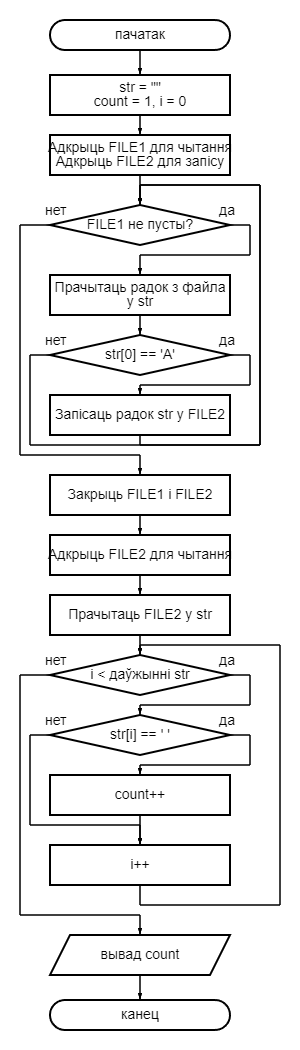
if (wordCount != -1) { //Калі -1 - то пры выкананні функцыі ўзнікла памылка

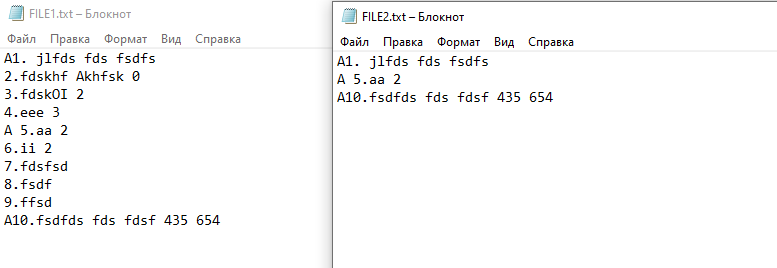
std::cout << "Колькасць слоў у FILE2: " << wordCount << std::endl;

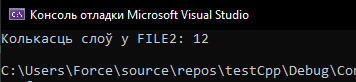
}

return 0;

}









#include <Windows.h>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Функцыя для запісу радка ў файл

void WriteStringIntoFile(const std::string& outputFile, std::string str);

// Функцыя для чытання файла і знаходжанні мінімальнай даўжыні элемента

int ReadAndProcess(const std::string& inputFile);

int main() {

SetConsoleOutputCP(1251);

std::string fileName = "myfile.txt";

std::string input;

// Уводзім радок

getline(cin, input);

// Запісваем

WriteStringIntoFile(fileName, input);

// Знаходзім

int x = ReadAndProcess(fileName);

if (x == -1) { //Калі была памылка адкрыцwя файла

return -1;

}

cout << x;

return 0;

}

void WriteStringIntoFile(const std::string& outputFile, std::string str) {

ofstream file(outputFile);

if (!file.is\_open() || file.fail()) {

cerr << "Не атрымалася адкрыць файл для запісу";

return;

}

file << str;

file.close();

}

int ReadAndProcess(const std::string& inputFile) {

ifstream file(inputFile);

if (!file.is\_open() || file.fail()) {

cerr << "Не атрымалася адкрыць файл для чытання";

return -1; // Вяртаем -1, каб пазначыць памылку

}

std::string str; // радок для групы сімвалаў

int minLength = INT\_MAX; // Выкарыстована максімальнае значэнне int у якасці пачатковага значэння

while (file >> str) { // Счытваем групы, падзеленыя прабеламі

if (minLength > str.length()) { // Атрымліваем даўжыню групы з дапамогай .length()

minLength = str.length();

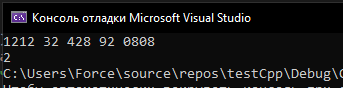
}

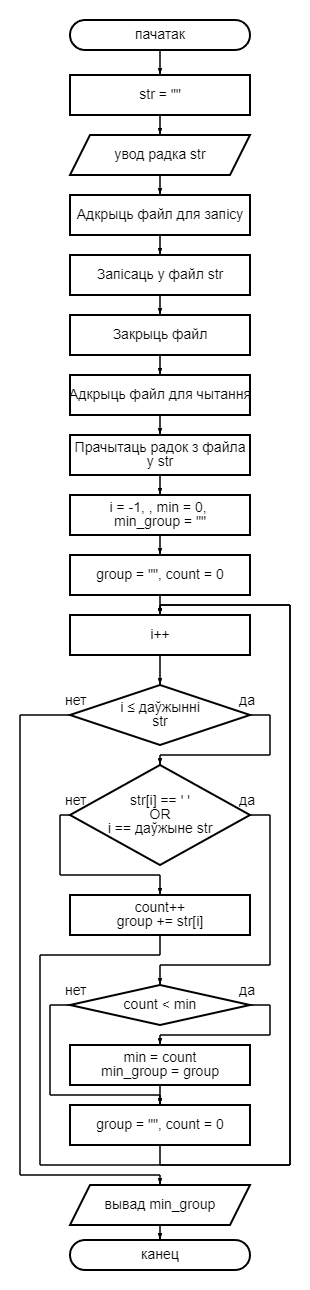
}

file.close();

return (minLength == INT\_MAX) ? 0 : minLength; // Вяртаем 0, калі не было знойдзена груп

}





Варыянт 4



#include <Windows.h>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Функцыя для праверкі, ці есць у радку лічбы

bool StringContainsDigits(const std::string& str);

// Функцыя для апрацоўкі зместа файла і запісу ў іншы

void ReadAndProcessFile(const std::string& fileName, const std::string& outputFileName);

// Функцыя для падліку радкооў, што пачынаюцца з сімвала с

int CountStringInFileStartsWith(const std::string& fileName, const char& c);

int main() {

SetConsoleOutputCP(1251);

ReadAndProcessFile("FILE1.txt", "filee22.txt"); //Апрацоўваем файл "FILE1.txt"

cout << CountStringInFileStartsWith("filee22.txt", 'A'); // Лічым колькасть радкоў, што пачынаюцца з A

return 0;

}

bool StringContainsDigits(const std::string& str) {

for (const char& c : str) { // Карстыюсь ітэратарам каб прайсці па ўсім сімвалам у радке

if ('0' <= c && c <= '9') { // Калі гэта лічба

return true;

}

}

return false;

}

void ReadAndProcessFile(const std::string& inputFileName, const std::string& outputFileName) {

std::ifstream inputFile(inputFileName);

std::ofstream outputFile(outputFileName);

//Праверка на паспяховае адкрыцце файла

if (!inputFile.is\_open() || inputFile.fail()) {

std::cerr << "Памылка адкрыцця файла " + inputFileName + " для чытання" << endl;

return;

}

//Праверка на паспяховае адкрыцце файла

if (!outputFile.is\_open() || outputFile.fail()) {

std::cerr << "Памылка адкрыцця файла " + outputFileName + " для запісу" << endl;

return;

}

std::string line; // Бягучы радок

while (getline(inputFile, line)) { // Атрымліваем радок файла

if (!StringContainsDigits(line)) { // Калі няма лічб

outputFile << line + '\n'; // Запісваем у новы файл

}

}

inputFile.close();

outputFile.close();

}

int CountStringInFileStartsWith(const std::string& fileName, const char& c) {

std::ifstream file(fileName);

if (!file.is\_open() || file.fail()) {

std::cerr << "Памылка адкрыцця файла " + fileName + " для чытання" << endl;

return -1;

}

std::string line;

int count = 0;

while (getline(file, line)) {

if (line[0] == c) { // Правяраяем з дапамогай індэксу каб першы сімвал быў роўны ўказанаму

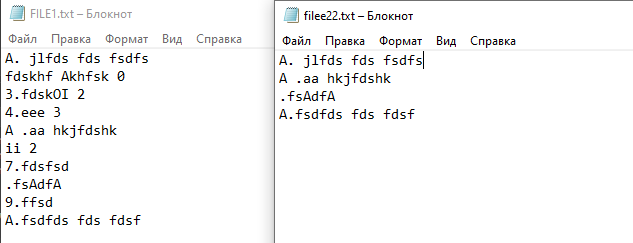
count++;

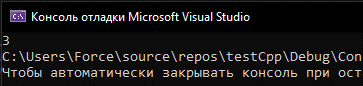
}

}

return count;

}







#include <Windows.h>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Функцыя для запісу радка ў файл

void WriteStringIntoFile(const std::string& outputFile, std::string str);

// Функцыя для чытання файла і знаходжанні мінімальнай даўжыні элемента

std::string ReadAndProcess(const std::string& inputFile);

bool containsDigits(std::string& str);

int main() {

SetConsoleOutputCP(1251);

std::string fileName = "myfile.txt";

std::string input;

// Уводзім радок

getline(cin, input);

// Запісваем

WriteStringIntoFile(fileName, input);

// Знаходзім

std:string x = ReadAndProcess(fileName);

if (x == "") { //Калі была памылка адкрыця файла або не было слоў в файле

return -1;

}

cout << x + "\n"<< x.length();

return 0;

}

void WriteStringIntoFile(const std::string& outputFile, std::string str) {

ofstream file(outputFile);

if (!file.is\_open() || file.fail()) {

cerr << "Не атрымалася адкрыць файл для запісу";

return;

}

file << str;

file.close();

}

std::string ReadAndProcess(const std::string& inputFile) {

ifstream file(inputFile);

if (!file.is\_open() || file.fail()) {

cerr << "Не атрымалася адкрыць файл для чытання";

return ""; // Вяртаем "", каб пазначыць памылку

}

std::string str; // радок для групы сімвалаў

std::string maxWord = ""; // найдаўжэйшае слова

int maxLength = 0;

while (file >> str) { // Счытваем групы, падзеленыя прабеламі

if (!containsDigits(str) && maxLength < str.length()) { // Калі няма лічб, то гэта слова. Атрымліваем даўжыню слова з дапамогай .length()

maxLength = str.length(); // Захоўваем даўжыню найдаўжэйшага слова

maxWord = str; // Захоўваем даўжэйшае слова

}

}

file.close();

return maxWord; // Вяртаем даўжэйшае слова

}

// функцыя праверкі ці ёсць лічбы ў ліку

bool containsDigits(std::string& str) {

for (const char& c : str) { // Канстантны ітэратар па радку str

if ('0' <= c && c <= '9') { // Калі лічба = вяртаем true

return true;

}

}

return false;

}

