Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра программной инженерии](https://www.belstu.by/fakultety/fit/vm)

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №9**

Дисциплина: Операционные системы

Тема: Файловая система

Выполнила:

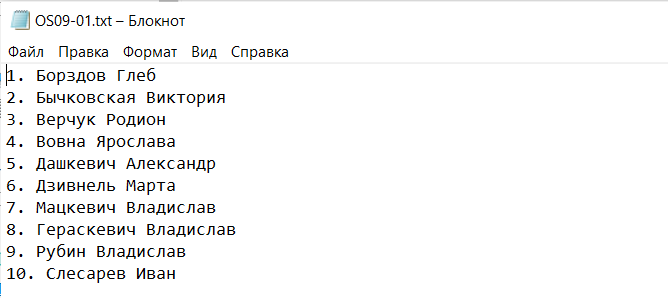
студентка 3 курса 5 группы

Вовна Ярослава Руслановна

Минск 2024

**Задание 01**

2)





#include <iostream>

#include <Windows.h>

#include <stdio.h>

#include <Shlwapi.h>

#pragma comment(lib, "Shlwapi.lib")

#define PATH L"D:\\лабы\\oc\\lab9\\OS09-01.txt"

using namespace std;

BOOL printFileInfo(LPWSTR fileName) {

if (!PathFileExists(fileName)) {

wprintf(L"File does not exist: %s\n", fileName);

return FALSE;

}

HANDLE hFile = CreateFile(fileName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile == INVALID\_HANDLE\_VALUE) {

wprintf(L"Error CreateFile: %s\n", fileName);

return FALSE;

}

BY\_HANDLE\_FILE\_INFORMATION lpFileInformation;

if (!GetFileInformationByHandle(hFile, &lpFileInformation)) {

wprintf(L"Error GetFileInformationByHandle: %s\n", fileName);

CloseHandle(hFile);

return FALSE;

}

const wchar\_t\* fileNameOnly = PathFindFileNameW(fileName);

wprintf(L"File Name: %s\n", fileNameOnly);

if (lpFileInformation.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) {

wprintf(L"File Type: folder\n");

}

else {

wprintf(L"File Type: file\n");

}

LARGE\_INTEGER fileSize;

fileSize.HighPart = lpFileInformation.nFileSizeHigh;

fileSize.LowPart = lpFileInformation.nFileSizeLow;

wprintf(L"File size: %lld byte\n", fileSize.QuadPart);

SYSTEMTIME creationTime;

FileTimeToSystemTime(&lpFileInformation.ftCreationTime, &creationTime);

wprintf(L"Date and time of creating: %02d.%02d.%d %02d:%02d:%02d\n",

creationTime.wDay, creationTime.wMonth, creationTime.wYear,

creationTime.wHour, creationTime.wMinute, creationTime.wSecond);

SYSTEMTIME lastWriteTime;

FileTimeToSystemTime(&lpFileInformation.ftLastWriteTime, &lastWriteTime);

wprintf(L"Date and time since last update: %02d.%02d.%d %02d:%02d:%02d\n",

lastWriteTime.wDay, lastWriteTime.wMonth, lastWriteTime.wYear,

lastWriteTime.wHour, lastWriteTime.wMinute, lastWriteTime.wSecond);

CloseHandle(hFile);

return TRUE;

}

BOOL printFileTxt(LPWSTR fileName) {

HANDLE hFile = CreateFile(fileName,GENERIC\_READ,FILE\_SHARE\_READ,NULL,OPEN\_EXISTING,FILE\_ATTRIBUTE\_NORMAL,NULL);

if (hFile == INVALID\_HANDLE\_VALUE) {

DWORD error = GetLastError();

wprintf(L"Error CreateFile: %s\n", fileName);

return FALSE;

}

DWORD bytesRead;

const DWORD bufferSize = 1024;

char buffer[bufferSize + 1];

wprintf(L"Text\n");

while (ReadFile(hFile, buffer, bufferSize, &bytesRead, NULL) && bytesRead > 0) {

buffer[bytesRead] = '\0';

printf("%s", buffer);

}

CloseHandle(hFile);

return TRUE;

}

int main() {

SetConsoleOutputCP(CP\_UTF8);

if (!printFileInfo((LPWSTR)PATH)) {

printf("Error printFileInfo\n");

}

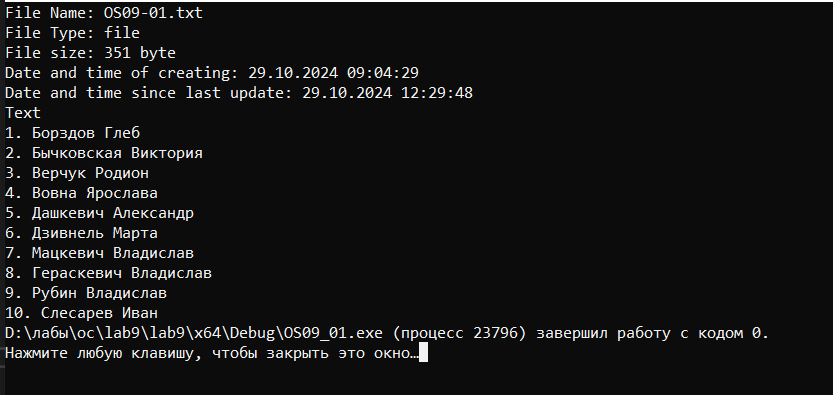
if (!printFileTxt((LPWSTR)PATH)) {

printf("Error printFileTxt\n");

}

return 0;

}



**Задание 02**

10-12)

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

#include <Shlwapi.h>

#pragma comment(lib, "Shlwapi.lib")

#define PATH L"D:\\лабы\\oc\\lab9\\OS09-02.txt"

using namespace std;

BOOL printFileTxt(LPWSTR fileName) {

HANDLE hFile = CreateFile(fileName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile == INVALID\_HANDLE\_VALUE) {

DWORD error = GetLastError();

wprintf(L"Error CreateFile: %s\n", fileName);

return FALSE;

}

DWORD bytesRead;

const DWORD bufferSize = 1024;

char buffer[bufferSize + 1];

wprintf(L"Text\n");

while (ReadFile(hFile, buffer, bufferSize, &bytesRead, NULL) && bytesRead > 0) {

buffer[bytesRead] = '\0';

printf("%s", buffer);

}

CloseHandle(hFile);

return TRUE;

}

BOOL delRowFileTxt(LPWSTR FileName, DWORD row) {

if (row <= 0) {

wprintf(L"row <= 0\n");

return false;

}

HANDLE fileHandle = CreateFile(FileName, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (fileHandle == INVALID\_HANDLE\_VALUE) {

wprintf(L"error Createfile\n");

return false;

}

LARGE\_INTEGER fileSize;

if (!GetFileSizeEx(fileHandle, &fileSize)) {

CloseHandle(fileHandle);

wprintf(L"error GetFileSizeEx\n");

return false;

}

unique\_ptr<char[]> buf(new char[fileSize.QuadPart + 1]);

DWORD bytesRead;

if (!ReadFile(fileHandle, buf.get(), fileSize.QuadPart, &bytesRead, NULL)) {

CloseHandle(fileHandle);

wprintf(L"error ReadFile\n");

return false;

}

buf[fileSize.QuadPart] = '\0';

string result;

int rowCount = 1;

char\* ptr = buf.get();

while (\*ptr) {

if (rowCount == row) {

while (\*ptr && \*ptr != '\n') {

ptr++;

}

if (\*ptr == '\n') ptr++;

rowCount++;

}

else {

while (\*ptr && \*ptr != '\n') {

result += \*ptr++;

}

result += '\n';

if (\*ptr == '\n') ptr++;

rowCount++;

}

}

SetFilePointer(fileHandle, 0, 0, FILE\_BEGIN);

DWORD bytesWritten;

if (!WriteFile(fileHandle, result.c\_str(), result.size(), &bytesWritten, NULL) ||

!SetEndOfFile(fileHandle)) {

CloseHandle(fileHandle);

wprintf(L"error WriteFile/SetEndOfFile\n");

return false;

}

CloseHandle(fileHandle);

return true;

}

int main() {

SetConsoleOutputCP(CP\_UTF8);

wprintf(L"Before:\n");

printFileTxt((LPWSTR)PATH);

wprintf(L"\n");

delRowFileTxt((LPWSTR)PATH, (DWORD)1);

delRowFileTxt((LPWSTR)PATH, (DWORD)3);

delRowFileTxt((LPWSTR)PATH, (DWORD)8);

delRowFileTxt((LPWSTR)PATH, (DWORD)10);

wprintf(L"\n");

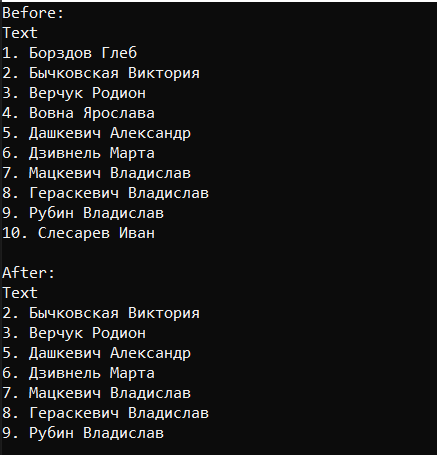
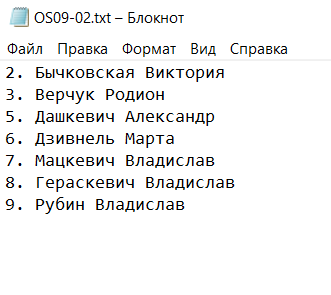
wprintf(L"After:\n");

printFileTxt((LPWSTR)PATH);

return 0;

}

13)

**Задание 03**

14-16)

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

#include <Shlwapi.h>

#include <cstdio>

#pragma comment(lib, "Shlwapi.lib")

#define PATH L"D:\\лабы\\oc\\lab9\\OS09-03.txt"

using namespace std;

BOOL printFileTxt(LPWSTR fileName) {

HANDLE hFile = CreateFile(fileName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile == INVALID\_HANDLE\_VALUE) {

DWORD error = GetLastError();

wprintf(L"Error CreateFile: %s\n", fileName);

return FALSE;

}

DWORD bytesRead;

const DWORD bufferSize = 1024;

char buffer[bufferSize + 1];

wprintf(L"Text\n");

while (ReadFile(hFile, buffer, bufferSize, &bytesRead, NULL) && bytesRead > 0) {

buffer[bytesRead] = '\0';

printf("%s", buffer);

}

CloseHandle(hFile);

return TRUE;

}

BOOL insRowFileTxt(LPWSTR fileName, LPWSTR str, DWORD row) {

HANDLE fileHandle = CreateFile(fileName, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (fileHandle == INVALID\_HANDLE\_VALUE) {

wprintf(L"error CreateFile\n");

return FALSE;

}

LARGE\_INTEGER fileSize;

if (!GetFileSizeEx(fileHandle, &fileSize)) {

CloseHandle(fileHandle);

wprintf(L"error GetFileSizeEx\n");

return FALSE;

}

int strLength = wcslen(str);

int bufferSize = WideCharToMultiByte(CP\_UTF8, 0, str, strLength, NULL, 0, NULL, NULL);

string insertStr(bufferSize, '\0');

WideCharToMultiByte(CP\_UTF8, 0, str, strLength, &insertStr[0], bufferSize, NULL, NULL);

string buf(fileSize.QuadPart, '\0');

DWORD bytesRead;

if (!ReadFile(fileHandle, &buf[0], fileSize.QuadPart, &bytesRead, NULL)) {

CloseHandle(fileHandle);

wprintf(L"error ReadFile\n");

return FALSE;

}

string result;

int rowCount = 0;

size\_t position = 0;

bool rowFound = false;

while (position < buf.size()) {

if (rowCount == row) {

result += insertStr + "\r\n";

rowFound = true;

}

while (position < buf.size() && buf[position] != '\n') {

result += buf[position++];

}

if (position < buf.size()) {

result += '\n';

position++;

}

rowCount++;

}

if (row == MAXDWORD) {

result += insertStr + "\r\n";

rowFound = true;

}

if (rowFound) {

SetFilePointer(fileHandle, 0, 0, FILE\_BEGIN);

DWORD bytesWritten;

if (!WriteFile(fileHandle, result.c\_str(), result.size(), &bytesWritten, NULL) ||

!SetEndOfFile(fileHandle)) {

CloseHandle(fileHandle);

wprintf(L"error WriteFile/SetEndOfFile\n");

return FALSE;

}

}

CloseHandle(fileHandle);

return TRUE;

}

int main() {

SetConsoleOutputCP(CP\_UTF8);

wprintf(L"Before:\n");

printFileTxt((LPWSTR)PATH);

wprintf(L"\n");

insRowFileTxt((LPWSTR)PATH, (LPWSTR)L"1. Борздов Глеб", 0);

insRowFileTxt((LPWSTR)PATH, (LPWSTR)L"4. Вовна Ярослава", -1);

insRowFileTxt((LPWSTR)PATH, (LPWSTR)L"10. Слесарев Иван", 5);

insRowFileTxt((LPWSTR)PATH, (LPWSTR)L"11. Шупенько Антон", 7);

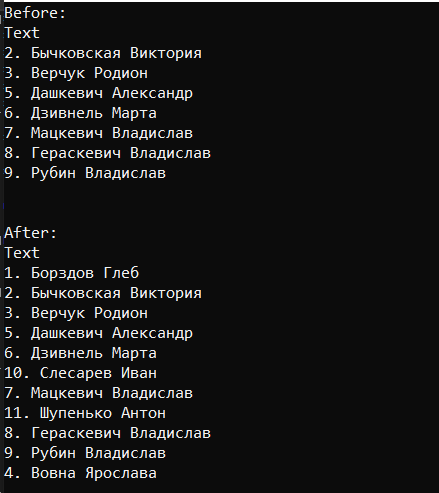
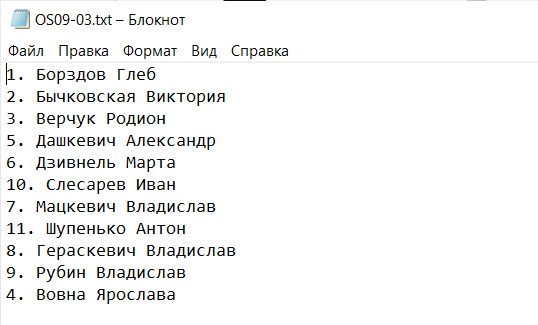
wprintf(L"\n");

wprintf(L"After:\n");

printFileTxt((LPWSTR)PATH);

return 0;

}

**Задание 04**

18-20)

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

#include <Shlwapi.h>

#include <cstdio>

#include <thread>

#pragma comment(lib, "Shlwapi.lib")

#define FILE\_PATH L"D:\\лабы\\oc\\lab9\\OS09-03.txt"

#define DIRECTORY\_PATH L"D:\\лабы\\oc\\lab9"

using namespace std;

int countLines(LPWSTR fileName) {

ifstream file(fileName);

if (!file) {

return 0;

}

int lineCount = 0;

string line;

while (getline(file, line)) {

++lineCount;

}

return lineCount;

}

void printWatchRowFileTxt(LPWSTR fileName, int mlsec) {

HANDLE hDir = CreateFile(DIRECTORY\_PATH, FILE\_LIST\_DIRECTORY, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL, OPEN\_EXISTING, FILE\_FLAG\_BACKUP\_SEMANTICS, NULL);

if (hDir == INVALID\_HANDLE\_VALUE) {

wprintf(L"error CreateFile\n");

return;

}

HANDLE hChange = FindFirstChangeNotification(DIRECTORY\_PATH, FALSE, FILE\_NOTIFY\_CHANGE\_SIZE | FILE\_NOTIFY\_CHANGE\_LAST\_WRITE);

if (hChange == INVALID\_HANDLE\_VALUE) {

wprintf(L"error FindFirstChangeNotification\n");

CloseHandle(hDir);

return;

}

int previousLineCount = countLines(fileName);

DWORD waitStatus;

while (mlsec > 0) {

waitStatus = WaitForSingleObject(hChange, 500);

if (waitStatus == WAIT\_OBJECT\_0) {

int currentLineCount = countLines(fileName);

if (currentLineCount != previousLineCount) {

wprintf(L"rows amount: %d >>> %d\n", previousLineCount, currentLineCount);

previousLineCount = currentLineCount;

}

FindNextChangeNotification(hChange);

}

mlsec -= 500;

}

FindCloseChangeNotification(hChange);

CloseHandle(hDir);

}

int main() {

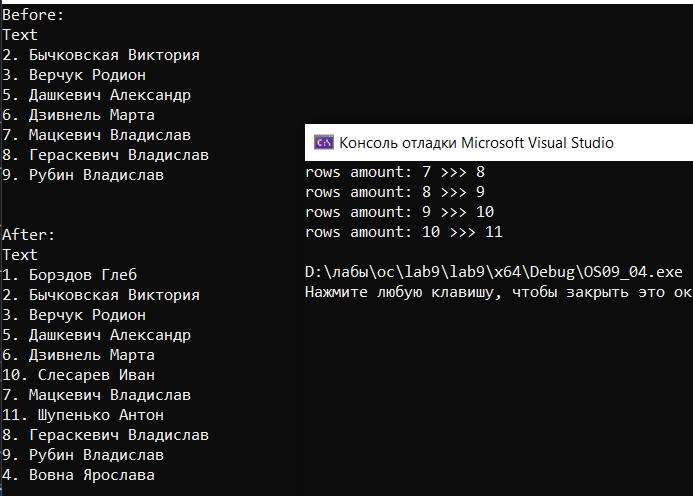
SetConsoleOutputCP(CP\_UTF8);

printWatchRowFileTxt((LPWSTR)FILE\_PATH, 10000);

return 0;

}

21)



**Задание 05**

22-23)

#include <stdio.h>

#include <stdlib.h>

int main() {

const char\* fileName = "OS09-05.txt";

FILE\* file = fopen(fileName, "r");

if (!file) {

perror("Error opening file");

return 0;

}

int lineCount = 0;

char ch;

while ((ch = fgetc(file)) != EOF) {

if (ch == '\n') {

lineCount++;

}

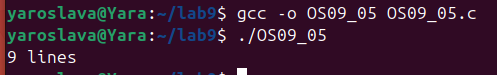
}

fclose(file);

printf("%d lines\n", lineCount);

return 0;

}



**Задание 06**

25-27)

#include <stdio.h>

#include <stdlib.h>

void sortLines(const char\* inputFile, const char\* outputFile, int isOdd) {

FILE\* file = fopen(inputFile, "r");

if (!file) {

perror("error fopen input");

return;

}

FILE\* output = fopen(outputFile, "w");

if (!output) {

perror("error output");

fclose(file);

return;

}

char line[256];

int lineNumber = 1;

while (fgets(line, sizeof(line), file)) {

if ((lineNumber % 2 == 1 && isOdd) || (lineNumber % 2 == 0 && !isOdd)) {

fputs(line, output);

}

lineNumber++;

}

fclose(file);

fclose(output);

}

int main(int argc, char\* argv[]) {

if (argc != 2) {

fprintf(stderr, "Usage: %s <integer>\n", argv[0]);

return 1;

}

int number = atoi(argv[1]);

const char\* inputFile = "OS09-05.txt";

if (number % 2 == 1) {

sortLines(inputFile, "OS09\_06\_1.txt", 1);

printf("created OS09\_06\_1.txt with odd lines.\n");

}

else {

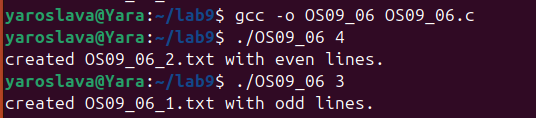
sortLines(inputFile, "OS09\_06\_2.txt", 0);

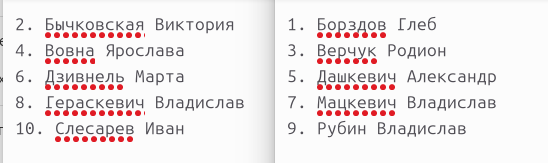
printf("created OS09\_06\_2.txt with even lines.\n");

}

return 0;

}





**Задание 07**

29-30)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#define FILENAME "OS09-07.txt"

int main() {

int fd = open(FILENAME, O\_RDONLY);

if (fd == -1) {

perror("Error opening file for reading");

return 1;

}

char buffer[351];

read(fd, buffer, sizeof(buffer));

printf("Full text:\n%s\n", buffer);

lseek(fd, 0, SEEK\_SET);

read(fd, buffer, 26);

buffer[26] = '\0';

printf("Read from beginning: %s\n", buffer);

lseek(fd, 28, SEEK\_SET);

read(fd, buffer, 3);

buffer[3] = '\0';

printf("Read from offset 28: %s\n", buffer);

lseek(fd,72, SEEK\_CUR);

read(fd, buffer, 10);

buffer[10] = '\0';

printf("Read from current position: %s\n", buffer);

close(fd);

return 0;

}

