Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №9**

Выполнил:

Студент 3 курса 4 группы ФИТ

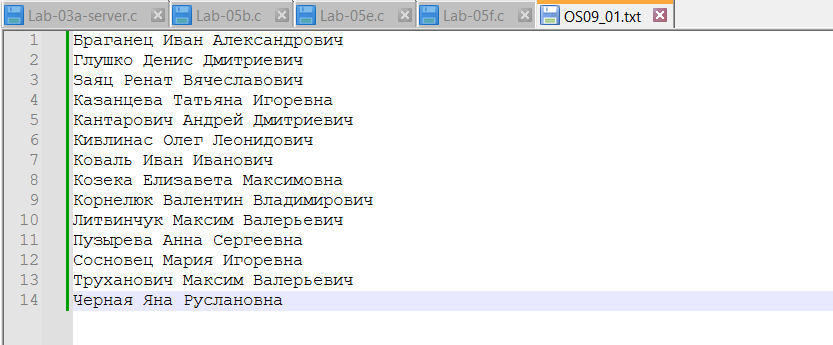
Сосновец Мария Игоревна

2024 г.

1. ***Внимание! Для работы с файловой системой использовать только OS API.***

**Задание 01.Windows**

1. С помощью Notepad (Notepad+) создайте на дисковом устройстве текстовый файл **OS09\_01.txt.** Заполните его 10 строками из списка студентов вашей подгруппы. Буквы кириллические.



1. Разработайте приложение **OS09\_01**.
2. Приложение **OS09\_01** вызывает функцию **printFileInfo**, имеющую следующий прототип.



1. Функция **printFileInfo** выводит в стандартный поток вывода следующую информацию:

- имя файла;

- тип файла;

- размер файла;

- дата и время создания файла;

- дата и время последнего обновления.

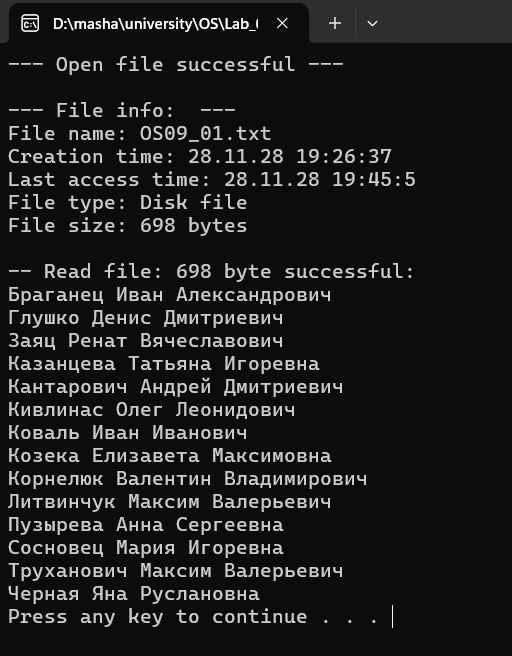
1. Приложение **OS09\_01** вызывает функцию **printFileTxt**, имеющую следующий прототип.



1. Функция **printFileTxt** выводит в стандартный поток вывода содержимое файла.
2. При вызове функции укажите в качестве параметра имя текстового файла, созданного в п.2.

Продемонстрируйте работоспособность приложения **OS09\_01**.

|  |
| --- |
| #define \_CRT\_NON\_CONFORMING\_WCSTOK  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <cstdlib>  #include "Windows.h"  #define FILE\_PATH L"D:/masha/university/OS/Lab\_09/OS09\_01/OS09\_01.txt"  using namespace std;  BOOL printFileInfo(LPWSTR FileName);  BOOL printFileTxt(LPWSTR FileName);  int main()  {  SetConsoleOutputCP(CP\_UTF8);  printFileInfo((LPWSTR)FILE\_PATH);  printFileTxt((LPWSTR)FILE\_PATH);  system("pause");  }  BOOL printFileInfo(LPWSTR FileName)  {  LPBY\_HANDLE\_FILE\_INFORMATION lpFileInformation = new BY\_HANDLE\_FILE\_INFORMATION();  DWORD fileType;  LPSYSTEMTIME systemTimeCreation = new SYSTEMTIME();  LPSYSTEMTIME systemTimeLastWrite = new SYSTEMTIME();  PLARGE\_INTEGER fileSize = new LARGE\_INTEGER();  try  {  HANDLE of = CreateFile(  FileName,  GENERIC\_WRITE | GENERIC\_READ,  NULL,  NULL,  OPEN\_ALWAYS,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (of == INVALID\_HANDLE\_VALUE)  {  throw "Open file failed";  }  wcout << L"--- Open file successful --- \n\n";  wcout << L"--- File info: --- \n";  // Конвертируем широкую строку в стандартную строку для обработки имени файла  wstring ws(FileName);  string filename(ws.begin(), ws.end());  const size\_t last\_slash\_idx = filename.find\_last\_of("\\/");  if (string::npos != last\_slash\_idx)  filename.erase(0, last\_slash\_idx + 1);  cout << "File name: " << filename << endl;  if (GetFileInformationByHandle(of, lpFileInformation))  {  if (FileTimeToSystemTime(&lpFileInformation->ftCreationTime, systemTimeCreation)  && FileTimeToSystemTime(&lpFileInformation->ftLastAccessTime, systemTimeLastWrite))  {  printf("Creation time: %u.%u.%u %u:%u:%u \n", systemTimeCreation->wDay, systemTimeCreation->wMonth, systemTimeCreation->wDay, systemTimeCreation->wHour, systemTimeCreation->wMinute, systemTimeCreation->wSecond);  printf("Last access time: %u.%u.%u %u:%u:%u \n", systemTimeLastWrite->wDay, systemTimeLastWrite->wMonth, systemTimeLastWrite->wDay, systemTimeLastWrite->wHour, systemTimeLastWrite->wMinute, systemTimeLastWrite->wSecond);  }  else  {  throw "FileTimeToSystemTime failed";  }  }  else  {  throw "GetFileInformationByHandle failed";  }  cout << "File type: ";  if (fileType = GetFileType(of))  {  switch (fileType)  {  case FILE\_TYPE\_CHAR: cout << "Printe, console file\n"; break;  case FILE\_TYPE\_DISK: cout << "Disk file\n"; break;  case FILE\_TYPE\_PIPE: cout << "Socket,named pipe, or anonymous pipe\n"; break;  case FILE\_TYPE\_REMOTE: cout << "UDEFINED\n"; break;  case FILE\_TYPE\_UNKNOWN: cout << "Unknown file type\n"; break;  }  }  else  {  throw "GetFileType failed";  }  if (GetFileSizeEx(of, fileSize))  {  cout << "File size: " << fileSize->QuadPart << " bytes" << endl;  }  else  {  throw "GetFileSizeEx failed";  }  CloseHandle(of);  }  catch (char\* err)  {  cout << "--- Error:\n" << err << endl;  return false;  }  return true;  }  BOOL printFileTxt(LPWSTR FileName)  {  LPBY\_HANDLE\_FILE\_INFORMATION lpFileInformation = new BY\_HANDLE\_FILE\_INFORMATION();  DWORD fileType;  PLARGE\_INTEGER fileSize = new LARGE\_INTEGER();  try  {  HANDLE of = CreateFile(  FileName,  GENERIC\_READ,  NULL,  NULL,  OPEN\_ALWAYS,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (of == INVALID\_HANDLE\_VALUE)  {  throw "Open file failed";  }  if (GetFileSizeEx(of, fileSize))  {  char\* buf = new char[fileSize->QuadPart];  ZeroMemory(buf, sizeof(buf));  DWORD n = NULL;  if (ReadFile(of, buf, fileSize->QuadPart, &n, NULL))  {  buf[fileSize->QuadPart] = '\0';  cout << "\n-- Read file: " << n << " byte successful: \n" << buf << endl;  }  }  else  {  throw "GetFileSizeEx failed";  }  CloseHandle(of);  }  catch (char\* err)  {  cout << "--- Error:\n" << err << endl;  return false;  }  return true;  } |



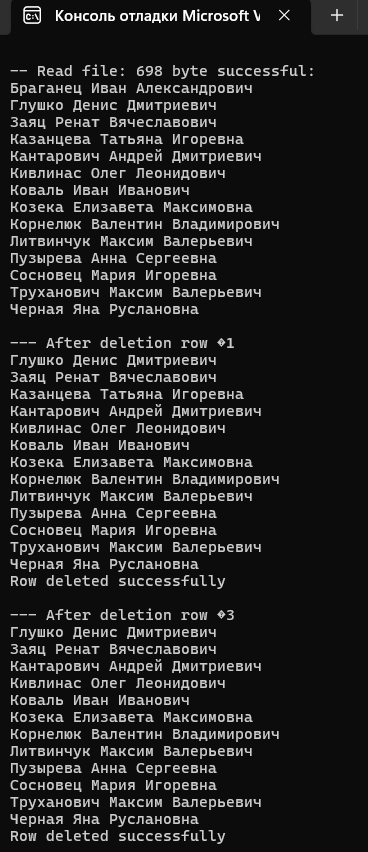
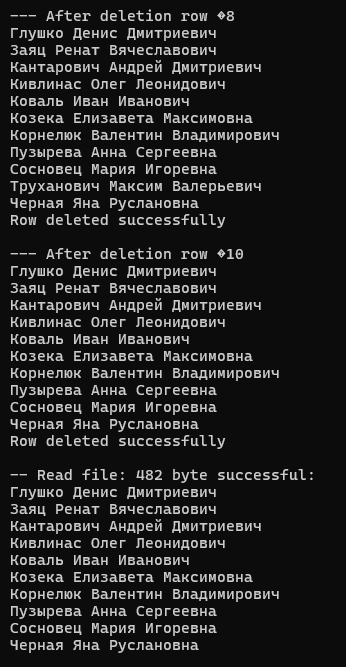
**Задание 02.Windows**

1. Разработайте приложение **OS09\_02.**
2. Приложение **OS09\_02** вызывает функцию **delRowFileTxt**, имеющую следующий прототип.



1. Функция применяется к файлу **OS09\_01.txt** (п.2) и вызывается последовательно 4 раза, с row = 1,3,8,10. Результат выполнения продемонстрируйте с помощью функции **printFileTxt** (п.6).
2. Продемонстрируйте работоспособность приложения **OS09\_02**.

|  |
| --- |
| 1. #define \_CRT\_NON\_CONFORMING\_WCSTOK 2. #define \_CRT\_SECURE\_NO\_WARNINGS 3. #include <iostream> 4. #include <cstdlib> 5. #include "Windows.h" 6. #define FILE\_PATH L"D:/masha/university/OS/Lab\_09/OS09\_01/OS09\_01.txt" 7. BOOL delRowFileTxt(LPWSTR FileName, DWORD row); 8. BOOL printFileTxt(LPWSTR FileName); 9. int main() 10. { 11. SetConsoleCP(CP\_UTF8); 12. SetConsoleOutputCP(CP\_UTF8); 13. LPWSTR fileName = (LPWSTR)FILE\_PATH; 14. printFileTxt(fileName); 15. delRowFileTxt(fileName, 1); 16. delRowFileTxt(fileName, 3); 17. delRowFileTxt(fileName, 8); 18. delRowFileTxt(fileName, 10); 19. printFileTxt(fileName); 20. } 21. BOOL delRowFileTxt(LPWSTR FileName, DWORD row) 22. { 23. PLARGE\_INTEGER fileSize = new LARGE\_INTEGER(); 24. int rowCount = 1; 25. int position = 0; 26. int positionAfter = 0; 27. bool rowFound = false; 28. try 29. { 30. if (row <= 0) 31. { 32. throw "Invalid number of raw"; 33. } 34. HANDLE of = CreateFile( 35. FileName, 36. GENERIC\_READ | GENERIC\_WRITE, 37. FILE\_SHARE\_READ, 38. NULL, 39. OPEN\_ALWAYS, 40. FILE\_ATTRIBUTE\_NORMAL, 41. NULL); 42. if (of == INVALID\_HANDLE\_VALUE) 43. { 44. CloseHandle(of); 45. throw "Open file failed"; 46. } 47. if (GetFileSizeEx(of, fileSize)) 48. {// Выделяем буферы для хранения содержимого файла и для записи после удаления 49. char\* buf = new char[fileSize->QuadPart + 1]; 50. char\* bufAfterDel = new char[fileSize->QuadPart + 1]; 51. ZeroMemory(buf, fileSize->QuadPart); 52. ZeroMemory(bufAfterDel, fileSize->QuadPart); 53. DWORD n = NULL; 54. if (ReadFile(of, buf, fileSize->QuadPart, &n, NULL)) 55. { 56. buf[fileSize->QuadPart] = '\0'; 57. while (buf[position] != '\0') 58. { 59. if (buf[position] == '\n') 60. { 61. rowCount++; 62. } 63. if (rowCount == row) 64. { 65. rowFound = true; 66. rowCount++; 67. position++; 68. while (buf[position] != '\n' && buf[position] != '\0') 69. { 70. position++; 71. } 72. if (row == 1) 73. { 74. position++; 75. } 76. } 77. bufAfterDel[positionAfter++] = buf[position++]; 78. if (buf[position] == '\0') 79. { 80. break; 81. } 82. } 83. bufAfterDel[positionAfter] = '\0'; 84. } 85. if (rowFound) 86. { 87. std::cout << "\n--- After deletion row №" << row << std::endl << bufAfterDel << std::endl; 88. if (SetFilePointer(of, 0, 0, FILE\_BEGIN) == 0) 89. { 90. if (WriteFile(of, bufAfterDel, strlen(bufAfterDel), &n, NULL)) 91. { 92. std::cout << "Row deleted successfully" << std::endl; 93. } 94. else 95. { 96. CloseHandle(of); 97. throw "WriteFile failed"; 98. } 99. if (!SetEndOfFile(of)) 100. { 101. CloseHandle(of); 102. throw "SetEndOfFile failed"; 103. } 104. } 105. else 106. { 107. CloseHandle(of); 108. throw "SetFilePointer failed"; 109. } 110. } 111. else 112. { 113. CloseHandle(of); 114. throw "Row isn't found"; 115. } 116. } 117. else 118. { 119. CloseHandle(of); 120. throw "GetFileSizeEx failed"; 121. } 122. CloseHandle(of); 123. } 124. catch (const char\* err) 125. { 126. std::cout << "--- Error:\n" << err << std::endl; 127. return false; 128. } 129. return true; 130. } 131. BOOL printFileTxt(LPWSTR FileName) 132. { 133. LPBY\_HANDLE\_FILE\_INFORMATION lpFileInformation = new BY\_HANDLE\_FILE\_INFORMATION(); 134. DWORD fileType; 135. PLARGE\_INTEGER fileSize = new LARGE\_INTEGER(); 136. try 137. { 138. HANDLE of = CreateFile( 139. FileName, 140. GENERIC\_READ, 141. NULL, 142. NULL, 143. OPEN\_ALWAYS, 144. FILE\_ATTRIBUTE\_NORMAL, 145. NULL); 146. if (of == INVALID\_HANDLE\_VALUE) 147. { 148. CloseHandle(of); 149. throw "Open file failed"; 150. } 151. if (GetFileSizeEx(of, fileSize)) 152. { 153. char\* buf = new char[fileSize->QuadPart]; 154. ZeroMemory(buf, sizeof(buf)); 155. DWORD n = NULL; 156. if (ReadFile(of, buf, fileSize->QuadPart, &n, NULL)) 157. { 158. buf[fileSize->QuadPart] = '\0'; 159. std::cout << "\n-- Read file: " << n << " byte successful: \n" << buf << std::endl; 160. } 161. else 162. { 163. CloseHandle(of); 164. throw "ReadFile failed"; 165. } 166. } 167. else 168. { 169. CloseHandle(of); 170. throw "GetFileSizeEx failed"; 171. } 172. CloseHandle(of); 173. } 174. catch (const char\* err) 175. { 176. std::cout << "--- Error:\n" << err << std::endl; 177. return false; 178. } 179. return true; 180. } |

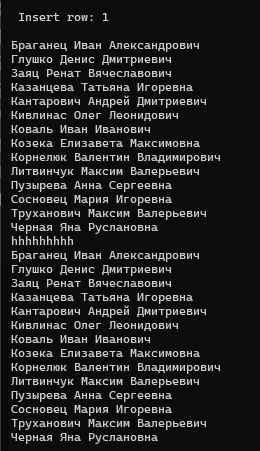
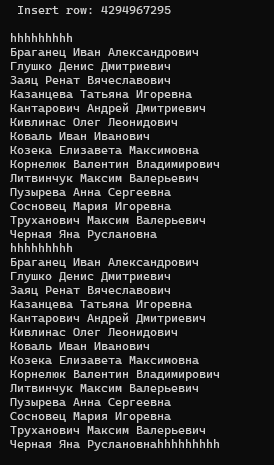
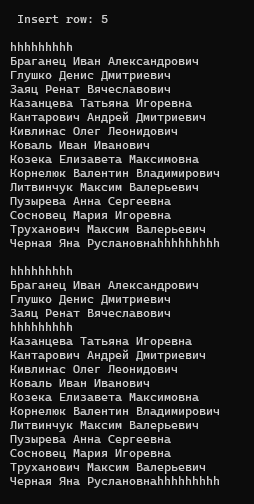
**Задание 03.Windows**

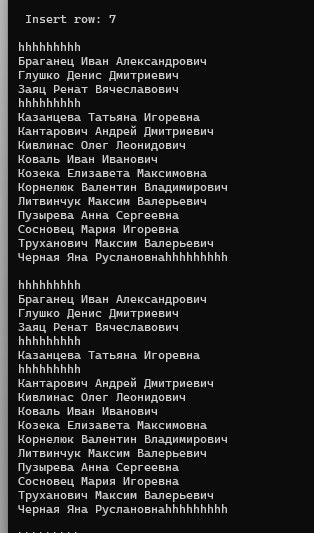
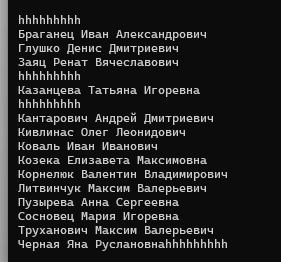
1. Разработайте приложение **OS09\_03.**
2. Приложение **OS09\_03** вызывает функцию **insRowFileTxt**, имеющую следующий прототип.

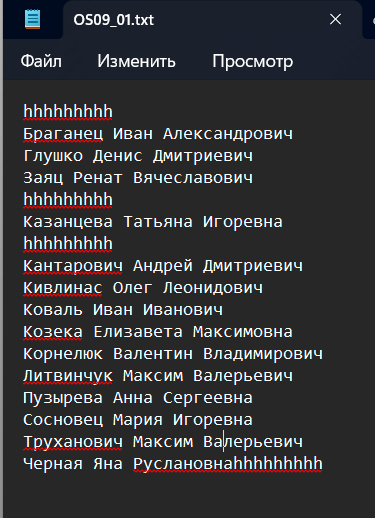


1. Функция применяется к файлу **OS09\_01.txt** (п.2) и вызывается последовательно 4 раза, с row = 0,-1,5,7. Результат выполнения продемонстрируйте с помощью функции **printFileTxt** (п.6).
2. Продемонстрируйте работоспособность приложения **OS09\_03**.

|  |
| --- |
| 1. #define \_CRT\_SECURE\_NO\_WARNINGS 2. #include <Windows.h> 3. #include <iostream> 4. #include <fileapi.h> 5. #define FILE\_PATH L"D:/masha/university/OS/Lab\_09/OS09\_01/OS09\_01.txt" 6. #define READ\_BYTES 1000 7. using namespace std; 8. BOOL printFileText(LPWSTR fileName) 9. { 10. try 11. { 12. HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); 13. if (hf == INVALID\_HANDLE\_VALUE) throw "[ERROR] Create or open file failed."; 14. DWORD n = NULL; 15. char buf[1024]; 16. ZeroMemory(buf, sizeof(buf)); 17. BOOL b = ReadFile(hf, &buf, READ\_BYTES, &n, NULL); 18. if (!b) throw "[ERROR] Read file failed"; 19. cout << buf << endl; 20. CloseHandle(hf); 21. return true; 22. } 23. catch (const char\* em) 24. { 25. cout << "[ERROR] " << em << endl; 26. return false; 27. } 28. } 29. BOOL insRowFileTxt(LPWSTR fileName, LPWSTR str, DWORD row) 30. { 31. char filepath[20]; 32. wcstombs(filepath, fileName, 20); 33. char stringToInsert[50]; 34. wcstombs(stringToInsert, str, 50); 35. cout << "\n Insert row: " << row << "\n\n"; 36. try 37. { 38. HANDLE hf = CreateFile(fileName, GENERIC\_READ, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); 39. if (hf == INVALID\_HANDLE\_VALUE) 40. { 41. CloseHandle(hf); 42. throw "[ERROR] Create or open file failed"; 43. } 44. DWORD n = NULL; 45. char buf[1024]; 46. BOOL b; 47. ZeroMemory(buf, sizeof(buf)); 48. b = ReadFile(hf, &buf, sizeof(buf), &n, NULL); 49. if (!b) 50. { 51. CloseHandle(hf); 52. throw ("[ERROR] Read file unsuccessful"); 53. } 54. if (!b) 55. { 56. cout << "Read file unsuccessful.\n"; 57. CloseHandle(hf); 58. return false; 59. } 60. cout << buf << endl; 61. CloseHandle(hf); 62. HANDLE hAppend = CreateFile(fileName, GENERIC\_WRITE, NULL, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL); 63. char editedBuf[1024]; 64. ZeroMemory(editedBuf, sizeof(editedBuf)); 65. int line = 1; 66. int j = 0; 67. int k = 0; 68. for (int i = 0; i < n; i++) 69. { 70. if (line == row) 71. { 72. for (int k = 0; k < sizeof(stringToInsert); k++) 73. { 74. editedBuf[j] = str[k]; 75. j++; 76. if (str[k + 1] == '\0') 77. { 78. editedBuf[j] = '\r'; 79. j++; 80. editedBuf[j] = '\n'; 81. j++; 82. row = 0; 83. break; 84. } 85. } 86. i--; 87. } 88. else 89. { 90. editedBuf[j] = buf[i]; 91. j++; 92. } 93. if (buf[i] == '\n') 94. line++; 95. // Проверка на конец файла для вставки в конец 96. if (buf[i + 1] == '\0' && row == -1) 97. { 98. for (int k = 0; k < sizeof(stringToInsert); k++) 99. { 100. editedBuf[j] = str[k]; 101. j++; 102. if (str[k + 1] == '\0') 103. { 104. editedBuf[j] = '\r'; 105. j++; 106. editedBuf[j] = '\n'; 107. j++; 108. row = 0; 109. break; 110. } 111. } 112. } 113. } 114. b = WriteFile(hAppend, editedBuf, j, &n, NULL); 115. if (!b) 116. { 117. CloseHandle(hAppend); 118. throw ("[ERROR] Write file unsuccessful\n"); 119. } 120. cout << editedBuf << endl; 121. CloseHandle(hAppend); 122. return true; 123. } 124. catch (const char\* em) 125. { 126. cout << em << " \n"; 127. return false; 128. } 129. } 130. int main() 131. { 132. setlocale(0, "ru"); 133. LPWSTR file = (LPWSTR)FILE\_PATH; 134. char str[] = "hhhhhhhhh"; 135. wchar\_t wStr[50]; 136. mbstowcs(wStr, str, strlen(str) + 1); 137. LPWSTR strToIns = wStr; 138. insRowFileTxt(file, strToIns, 1); 139. insRowFileTxt(file, strToIns, -1); 140. insRowFileTxt(file, strToIns, 5); 141. cout << '\n'; 142. insRowFileTxt(file, strToIns, 7); 143. printFileText(file); 144. } |



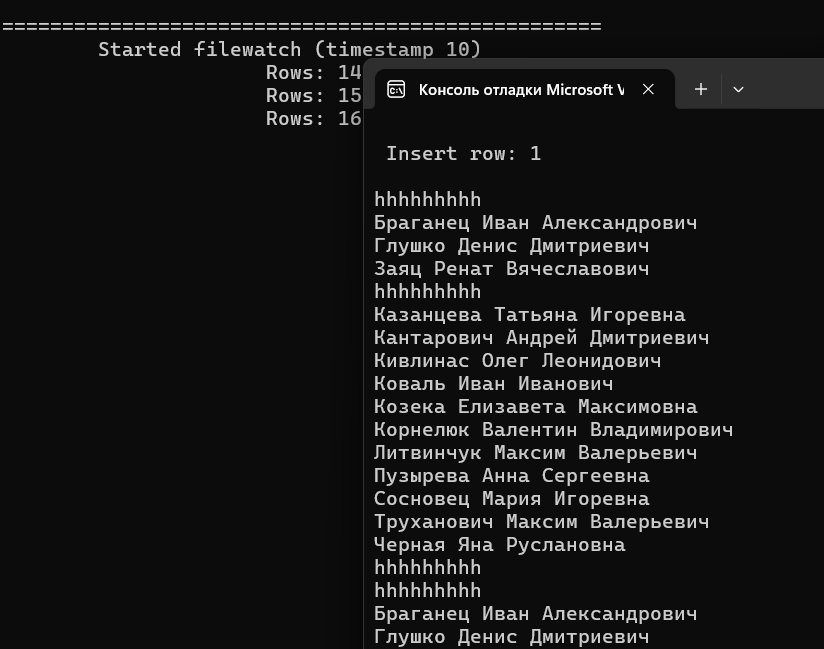
**Задание 04.Windows**

1. Разработайте приложение **OS09\_04.**
2. Приложение **OS09\_04** вызывает функцию **printWathRowFileTxt**, имеющую следующий прототип.



1. Функция применяется к файлу **OS09\_01.txt** (п.2), следит (***используйте функцию наблюдения за файлами в каталоге***) за изменением количества строк в файле в течении mlsec и выводит информацию об изменениях в стандартный поток вывода.
2. Продемонстрируйте работоспособность приложения **OS09\_04** совместно с приложениями **OS09\_03** и **OS09\_04**.

|  |
| --- |
| #define \_CRT\_NON\_CONFORMING\_WCSTOK  #define \_CRT\_SECURE\_NO\_WARNINGS  using namespace std;  #include <iostream>  #include <cstdlib>  #include "Windows.h"  #define FILE\_PATH L"D:/masha/university/OS/Lab\_09/OS09\_01/OS09\_01.txt"  #define DIR\_PATH L"D:/masha/university/OS/Lab\_09/OS09\_01"  int rowC = 0;  BOOL printWatchRowFileTxt(LPWSTR FileName, DWORD mlsec)  {  PLARGE\_INTEGER fileSize = new LARGE\_INTEGER();  LPWSTR path = (LPWSTR)DIR\_PATH;  char\* cFileName = new char[wcslen(FileName) \* sizeof(char) + 1];  wcstombs(cFileName, FileName, strlen(cFileName));  int i = 0;  int lenName = 0;  try  {  HANDLE notif = FindFirstChangeNotification(path, false, FILE\_NOTIFY\_CHANGE\_LAST\_WRITE);  DWORD err = GetLastError();  clock\_t t1 = clock();  clock\_t t2 = clock();  DWORD dwWaitStatus;  cout << "\n==================================================\n";  printf("\tStarted filewatch (timestamp %d)", t1);  while (true)  {  dwWaitStatus = WaitForSingleObject(notif, mlsec);  switch (dwWaitStatus)  {  case WAIT\_OBJECT\_0:  {  if (FindNextChangeNotification(notif) == FALSE)  break;  else  {  int position = 0;  int rowCount = 0;  HANDLE of = CreateFile(  FileName,  GENERIC\_READ,  FILE\_SHARE\_READ,  NULL,  OPEN\_ALWAYS,  FILE\_ATTRIBUTE\_NORMAL,  NULL);  if (of == INVALID\_HANDLE\_VALUE)  {  CloseHandle(of);  throw "[ERROR] Open file failed";  }  if (GetFileSizeEx(of, fileSize))  {  char\* buf = new char[(fileSize->QuadPart + 1) \* sizeof(char)];  ZeroMemory(buf, (fileSize->QuadPart + 1) \* sizeof(char));  DWORD n = NULL;  if (ReadFile(of, buf, fileSize->QuadPart, &n, NULL))  while (buf[position++] != '\0')  if (buf[position] == '\n')  rowCount++;  }  if (rowC != rowCount)  {  printf("\n\t\t Rows: %d", rowCount);  rowC = rowCount;  }  CloseHandle(of);  }  }  }  t2 = clock();  if (t2 - t1 > mlsec)  break;  }  CloseHandle(notif);  printf("\n\tEnded filewatch (timestamp %d)\n", t2);  }  catch (const char\* err)  {  cout << "[ERROR] " << err << "\n";  return false;  }  return true;  }  int main()  {  setlocale(LC\_ALL, "ru");  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  LPWSTR fileName = (LPWSTR)FILE\_PATH;  printWatchRowFileTxt(fileName, 30000);  } |



**Задание 05.Linux**

1. Создайте текстовый файл **OS09\_05.txt,** аналогичный файлу **OS09\_01.txt** (п.2).
2. Разработайте приложение **OS09\_05,** подсчитывающее количество строк и выводящее это значение в стандартный поток.
3. Продемонстрируйте работоспособность приложения **OS09\_05**.

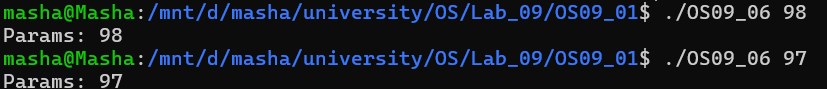
|  |
| --- |
| 1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <sys/types.h> 4. #include <time.h> 5. #include <unistd.h> 6. #include <sys/stat.h> 7. #include <fcntl.h> 8. #include <locale.h> 9. int main() 10. { 11. char\* locale = setlocale(LC\_ALL, ""); 12. char c; 13. int rowNumber = 1; 14. int in = open("OS09\_05.txt", O\_RDONLY); 15. while (read(in, &c, 1) == 1) 16. { 17. if (c == '\n') 18. { 19. rowNumber++; 20. } 21. } 22. printf("Rows quantity: %d \n", rowNumber); 23. return 0; 24. } |



**Задание 06.Linux**

1. Разработайте приложение **OS09\_06,** принимающее 1 параметр, принимающее целочисленное числовое значение.
2. Если числовое значение принимает нечетное значение, то приложение создает новый файл **OS09\_06\_1.txt**, содержащий только нечетные строки из файла **OS09\_05.txt**.
3. Если числовое значение принимает четное значение, то приложение создает новый файл **OS09\_06\_2.txt**, содержащий только четные строки из файла **OS09\_05.txt**.
4. Продемонстрируйте работоспособность приложения **OS09\_06**.

|  |
| --- |
| 1. #define \_POSIX\_C\_SOURCE 199309L 2. #include <stdio.h> 3. #include <stdlib.h> 4. #include <sys/types.h> 5. #include <time.h> 6. #include <unistd.h> 7. #include <sys/stat.h> 8. #include <fcntl.h> 9. #include <locale.h> 10. int main(int argc, char\* argv[]) 11. { 12. int number; 13. char\* locale = setlocale(LC\_ALL, "ru\_RU.UTF-8"); 14. char c[1]; 15. int in = open("Task5.txt", O\_RDONLY); 16. int out; 17. \_Bool evenNumber = 0; 18. if (argc != 2) 19. { 20. printf("One number argument is required: odd number - odd rows, even number - even rows\n"); 21. exit(1); 22. } 23. number = atoi(argv[1]); 24. printf("Params: %d\n", number); 25. if (number % 2 != 0) 26. { 27. out = open("Task6\_1.txt", O\_RDWR | O\_CREAT); 28. } 29. else 30. { 31. evenNumber = 1; 32. out = open("Task6\_2.txt", O\_RDWR | O\_CREAT); 33. } 34. while (read(in, &c, 1) == 1) 35. { 36. if (evenNumber == 0) 37. { 38. write(out, &c, 1); 39. } 40. if (\*c == '\n') 41. { 42. evenNumber = !evenNumber; 43. } 44. } 45. exit(0); 46. } |



**Задание 07.Linux**

1. Разработайте приложение **OS09\_07,** демонстрирующее возможности функции **lseek.**
2. Поясните назначение и принцип работы функции **lseek**.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <time.h>  #include <unistd.h>  #include <sys/stat.h>  #include <fcntl.h>  #include <locale.h>  #include <stdbool.h>  int main(int argc, char\* argv[])  {  char\* locale = setlocale(LC\_ALL, "");  int in;  long pos, offs;  char buffer[10];  in = open("OS09\_05.txt", O\_RDONLY);  offs = 0L;  pos = lseek(in, offs, SEEK\_SET);  printf("SEEK\_SET, offset=0L: %ld\n", pos);  offs = 10L;  read(in, buffer, 10);  printf("10 chars read: %s\n", buffer);  pos = lseek(in, offs, SEEK\_CUR);  printf("SEEK\_CUR, offset=10L: = %ld\n", pos);  offs = 20L;  pos = lseek(in, offs, SEEK\_END);  printf("SEEK\_END, offset=20L: = %ld\n", pos);  close(in);  return 0;  } |

