***Министерство образования Республики Беларусь***

***Учреждение Образования***

***«Брестский Государственный Технический Университет»***

***Кафедра ИИТ***

**Лабораторная работа №12**

**По дисциплине ОАиП за II семестр**

**Тема: «Текстовые файлы»**

**Выполнил:**

Студент 1-го курса

Группы ИИ-15(1)

Волк И. А.

**Проверила:**

Хацкевич М. В.

Брест 2018

Цель: научиться работать с текстовыми файлами.

**Задание 1**

Реализовать меню, разработанное с применением массива указателей на функции.

**Задание 2**

Предварительно подготовить(программно или с помощью текстового редактора) текстовый файл,содержимым которого является массив (матрица) из чисел. В соответствии с индивидуальным заданием написать программу обработки содержимого файла.

Для каждого из вариантов задания необходимо выполнить следующие действия. Просмотреть содержимое исходного файла. Считать содержимое файла в одномерный (двумерный) динамический массив. Обработать

динамический массив согласно варианту задания. Полученный результат записать в конец исходного файла.

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <fstream>

#include <conio.h>

#include <string>

#include <cmath>

#include <string>

using namespace std;

int menu();

template<typename T>

T\* increase(T\*, int\*);

void error(const char\*, int);

int \* stringToIntArray(char\*, int);

int strToInt(char \*);

void writeToFile(const char\*, const char\*, int);

void writeToFile(const char\*, const int\*, int, int);

void writeMatrixToFile(const char\*, int&, int&);

void outMatrixFromFile(const char\*, int&, int&);

void sortMatrix(int\*\*, int, int);

void insertSort(int \*, int);

void sortMatrixFromFile(const char\*, int&, int&);

void searchElFromFile(const char\*, int &, int &);

int \*\* readMatrixFromFile(const char\*, int, int, bool);

int getMenuItem(int, int);

bool isResult(const char\*, int);

void finishProgram();

char \* getNameOfFile();

void(\*function[])(const char\*, int&, int&) = { writeMatrixToFile,

outMatrixFromFile,

sortMatrixFromFile,

searchElFromFile };

int main()

{

int numOfLines = 0, numOfColumns = 0;

char \*name = getNameOfFile();

char nameOfFile[1024];

strcpy\_s(nameOfFile, sizeof(nameOfFile), name);

writeMatrixToFile(nameOfFile, numOfLines, numOfColumns);

while (1)

{

int i = menu();

if (i == 1)

name = getNameOfFile();

else if (i > 1 && i < 6)

function[i - 2](nameOfFile, numOfLines, numOfColumns);

else

finishProgram();

}

\_getch();

return 0;

}

int menu()

{

cout << "1. Set name of file" << endl;

cout << "2. Write to file" << endl;

cout << "3. Read from file" << endl;

cout << "4. Sort the file" << endl;

cout << "5. Search the element" << endl;

cout << "6. Exit" << endl;

int input = getMenuItem(1, 6);

system("cls");

return input;

}

template<typename T>

T\* increase(T\* arr, int \*size)

{

(\*size)++;

int sizeVar = (\*size);

T\* arrRes = new T[sizeVar];

for (int i = 0; i < sizeVar - 1; i++)

arrRes[i] = arr[i];

return arrRes;

}

void error(const char\* content, int timer)

{

system("cls");

cout << content << endl;

Sleep(timer);

system("cls");

}

void writeToFile(const char\* nameOfFile, const char\* content, int toTheEnd)

{

ofstream fout;

if(toTheEnd)

fout.open(nameOfFile, ios::out | ios::app);

else

fout.open(nameOfFile, ios::out);

if (!fout)

{

error("Cannot open the file!", 500);

return;

}

fout << content;

fout.close();

}

void writeToFile(const char\* nameOfFile, const int\* content, int sizeOfContent, int toTheEnd)

{

ofstream fout;

if (toTheEnd)

fout.open(nameOfFile, ios::out | ios::app);

else

fout.open(nameOfFile, ios::out);

if (!fout)

{

error("Cannot open the file!", 500);

return;

}

for(int i = 0 ; i < sizeOfContent; i++)

fout << content[i] << " ";

fout.close();

}

void writeMatrixToFile(const char\* nameOfFile, int &numOfLines, int &numOfColumns)

{

system("cls");

//int numOfLines, numOfColumns;

cout << "Number of lines: "; cin >> numOfLines;

cout << "Number of columns: "; cin >> numOfColumns;

system("cls");

writeToFile(nameOfFile, "", 0);

for (int i = 0; i < numOfLines; i++)

{

int \* line = new int[numOfColumns];

for (int j = 0; j < numOfColumns; j++)

{

int num;

cout << i + 1 << j + 1 << ": ";

cin >> num;

line[j] = num;

}

cout << endl;

writeToFile(nameOfFile, line, numOfColumns, 1);

writeToFile(nameOfFile, "\n", 1);

}

system("cls");

}

int \*\* readMatrixFromFile(const char\* nameOfFile, int numOfLines, int numOfColumns, bool res)

{

ifstream fcin;

fcin.open(nameOfFile);

if (!fcin)

{

error("Cannot open the file!", 500);

return NULL;

}

int \*\* intArr = new int\*[numOfLines];

for (int i = 0; i < numOfLines; i++)

intArr[i] = new int[numOfColumns];

char \*\* chArr = new char\*[numOfLines];

for (int i = 0; i < numOfLines; i++)

chArr[i] = new char[100];

int j = 0;

for (int i = 0; (res) ? (i < (numOfLines \* 2 + 1)) : (i < numOfLines); i++)

{

if (res&&i <= numOfLines)

{

string s;

getline(fcin, s);

continue;

}

if (i > numOfLines)

{

string s;

getline(fcin, s);

strcpy\_s(chArr[j], sizeof(char) \* 100, s.c\_str());

j++;

}

else

{

string s;

getline(fcin, s);

strcpy\_s(chArr[i], sizeof(char) \* 100, s.c\_str());

}

}

for(int i = 0; i < numOfLines; i++)

intArr[i] = stringToIntArray(chArr[i], numOfColumns);

return intArr;

}

void outMatrixFromFile(const char\* nameOfFile, int &numOfLines, int &numOfColumns)

{

cout << " 1. Matrix \n 2. Result";

int item = getMenuItem(1, 2) - 1;

system("cls");

if (item && !isResult(nameOfFile, numOfLines))

{

error("The result does not yet exist", 500);

return;

}

int \*\* matrix = readMatrixFromFile(nameOfFile, numOfLines, numOfColumns, item);

if(item) cout << "Result from file: " << endl;

else cout << "Matrix from file: " << endl;

for (int i = 0; i < numOfLines; i++)

{

for (int j = 0; j < numOfColumns; j++)

cout << "\t" << matrix[i][j];

cout << endl;

}

system("pause");

system("cls");

}

int getMenuItem(int a, int b)

{

int res;

while (1)

{

res = \_getch() - 48;

if (res >= a&&res <= b)

break;

}

return res;

}

int \* stringToIntArray(char\* string, int numOfEl)

{

int \* el = new int[numOfEl], j = 0, num = 0;

char elNow[100] = "";

for (int i = 0; string[i] != '\0'; i++)

{

if (string[i] == ' ')

{

if (j != 0)

{

elNow[j] = '\0';

el[num] = strToInt(elNow);

num++;

}

strcpy\_s(elNow, sizeof(elNow), "");

j = 0;

}

else if (string[i + 1] == '\0')

{

if (j != 0)

{

el[num] = strToInt(elNow);

num++;

}

}

else

{

elNow[j] = string[i];

j++;

}

}

return el;

}

int strToInt(char\* a)

{

int res = 0;

int len = strlen(a);

for (int i = 0, j = len - 1; i < len; i++, j--)

{

int num = (int)a[j] - 48;

res += num \* (int)pow(10.0, (double)i);

}

return res;

}

void sortMatrix(int\*\* matrix, int numOfLines, int numOfColumns)

{

for (int i = 0; i < numOfLines; i++)

insertSort(matrix[i], numOfColumns);

}

void insertSort(int \* arr, int size)

{

for (int i = 1; i < size; i++)

{

int temp = arr[i];

int j;

for (j = i; arr[j - 1] > temp && j != 0; j--)

arr[j] = arr[j - 1];

arr[j] = temp;

}

}

bool isResult(const char\* nameOfFile, int numOfLines)

{

ifstream fcin;

fcin.open(nameOfFile);

string s;

int i;

for (i = -1; !fcin.eof(); i++)

{

getline(fcin, s);

if (!strcmp(s.c\_str(), "r"))

return true;

}

return false;

}

void sortMatrixFromFile(const char\* nameOfFile, int &numOfLines, int &numOfColumns)

{

if (isResult(nameOfFile, numOfLines))

{

error("Matrix is already sorted!", 500);

return;

}

int \*\* matrix = readMatrixFromFile(nameOfFile, numOfLines, numOfColumns, 0);

sortMatrix(matrix, numOfLines, numOfColumns);

writeToFile(nameOfFile, "r\n", 1);

for (int i = 0; i < numOfLines; i++)

{

writeToFile(nameOfFile, matrix[i], numOfColumns, 1);

writeToFile(nameOfFile, "\n", 1);

}

cout << "Complate!" << endl;

Sleep(500);

system("cls");

return;

}

void searchElFromFile(const char\* nameOfFile, int &numOfLines, int &numOfColumns)

{

if (!isResult(nameOfFile, numOfLines))

{

error("File is not sorted!", 500);

return;

}

int el;

cout << "Element: "; cin >> el;

system("cls");

int \*\* matrix = readMatrixFromFile(nameOfFile, numOfLines, numOfColumns, 1);

cout << "Id: ";

for (int i = 0; i < numOfLines; i++)

{

int shift = numOfColumns / 2;

int pos = shift;

do

{

if (matrix[i][pos] == el)

cout << "(" << i << "; " << pos << ") ";

else if (el < matrix[i][pos])

{

shift /= 2;

pos = shift;

}

else if (el > matrix[i][pos])

{

shift /= 2;

if (!shift)

{

pos++;

if(matrix[i][pos] == el)

cout << "(" << i << "; " << pos << ") ";

}

else pos += shift;

}

} while (shift != 0);

}

system("pause");

system("cls");

}

void finishProgram()

{

cout << "Good lock!" << endl;

Sleep(500);

exit(0);

}

char \* getNameOfFile()

{

char nameOfFile[1024];

cout << "Name of file: ";

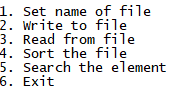
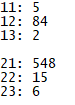
gets\_s(nameOfFile);

system("cls");

return nameOfFile;

}

Результат:



Search:



**Задание 3** **«Основные операции с текстовыми файлами»**

19. Дан текстовый файл. Заменить в нем все прописные русские буквы на строчные, а все строчные — на прописные.

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <cmath>

#include <string>

#include <fstream>

#include <conio.h>

using namespace std;

void error(const char\*, int);

char \*\* readFromFile(char\*, int&);

void writeToFile(const char\*, const char\*, int);

int numOfLines(char\*);

void inverseCase(char\*);

int main()

{

setlocale(LC\_ALL, "rus");

while (1)

{

int numOfLines;

char \*\* text = readFromFile("text.txt", numOfLines);

writeToFile("text.txt", "", 0);

for (int i = 0; i < numOfLines; i++)

{

inverseCase(text[i]);

writeToFile("text.txt", text[i], 1);

writeToFile("text.txt", "\n", 1);

}

for (int i = 0; i < numOfLines; i++)

cout << text[i] << endl;

int input; bool br = false;

while (1)

{

input = \_getch();

if (input == 32)

break;

else if (input == 27)

{

br = true;

break;

}

}

if (br)

break;

system("cls");

}

system("pause");

return 0;

}

char \*\* readFromFile(char\* nameOfFile, int& numOfL)

{

ifstream fcin;

fcin.open(nameOfFile);

if (!fcin)

{

error("Cannot open the file!", 1000);

char \*\* error = new char\*;

return error;

}

numOfL = numOfLines(nameOfFile);

char \*\* lines = new char\*[numOfL];

for (int i = 0; i < numOfL; i++)

lines[i] = new char[100];

int i = 0;

string s;

for(; getline(fcin, s); i++)

strcpy\_s(lines[i], sizeof(s), s.c\_str());

return lines;

}

void error(const char\* content, int timer)

{

system("cls");

cout << content << endl;

Sleep(timer);

system("cls");

}

int numOfLines(char\* nameOfFile)

{

ifstream fcin(nameOfFile);

int numOfLines = 0;

string s;

while (getline(fcin, s))

numOfLines++;

return numOfLines;

}

void writeToFile(const char\* nameOfFile, const char\* content, int toTheEnd)

{

ofstream fout;

if (toTheEnd)

fout.open(nameOfFile, ios::out | ios::app);

else

fout.open(nameOfFile, ios::out);

if (!fout)

{

error("Cannot open the file!", 500);

return;

}

fout << content;

fout.close();

}

void inverseCase(char\* str)

{

char lower[] = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

char upper[] = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

int size = sizeof(lower) / sizeof(\*lower);

int i = 0;

while (str[i] != '\0')

{

int j = 0;

while (str[i] != upper[j] && j != size)

j++;

if (j == size)

{

j = 0;

while (str[i] != lower[j] && j != size)

j++;

if (j != size)

str[i] = upper[j];

}

else

{

str[i] = lower[j];

}

i++;

}

}

Результат выполнения:



**Задание 4«Анализ и форматирование текста»**

Дан символ C — прописная (заглавная) русская буква и текстовый файл. Создать строковый файл и записать в него все слова из исходного файла, начинающиеся на эту букву (прописную или строчную). Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки. Если исходный файл не содержит подходящих слов, то оставить результирующий файл пустым.

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <cmath>

#include <string>

#include <fstream>

#include <conio.h>

using namespace std;

void error(const char\*, int);

char \*\* readFromFile(char\*, int&);

int numOfLines(char\*);

char \*\* getWords(char\*, int&);

template<typename T>

T\* increase(T\*, int&);

void writeToFile(const char\*, const char\*, int);

int main()

{

setlocale(LC\_ALL, "rus");

char C = 'а';

int numOfLines;

char \*\* text = readFromFile("initial.txt", numOfLines);

writeToFile("result.txt", "", 0);

for (int i = 0; i < numOfLines; i++)

{

int numOfWords;

char \*\* words = getWords(text[i], numOfWords);

for (int j = 0; j < numOfWords; j++)

if (words[j][0] == C)

{

writeToFile("result.txt", words[j], 1);

writeToFile("result.txt", " ", 1);

}

}

system("pause");

return 0;

}

char \*\* readFromFile(char\* nameOfFile, int& numOfL)

{

ifstream fcin;

fcin.open(nameOfFile);

if (!fcin)

{

error("Cannot open the file!", 1000);

char \*\* error = new char\*;

return error;

}

numOfL = numOfLines(nameOfFile);

char \*\* lines = new char\*[numOfL];

for (int i = 0; i < numOfL; i++)

lines[i] = new char[100];

int i = 0;

string s;

for (; getline(fcin, s); i++)

strcpy\_s(lines[i], sizeof(s), s.c\_str());

return lines;

}

int numOfLines(char\* nameOfFile)

{

ifstream fcin(nameOfFile);

int numOfLines = 0;

string s;

while (getline(fcin, s))

numOfLines++;

return numOfLines;

}

void error(const char\* content, int timer)

{

system("cls");

cout << content << endl;

Sleep(timer);

system("cls");

}

char \*\* getWords(char\* text, int& numOfWords)

{

numOfWords = 0;

char \*\* words = new char\*;

char wordNow[1024] = "";

int i = 0, chInWord = 0;

while (text[i] != '\0')

{

if (text[i] == ' ')

{

if (chInWord != 0)

{

wordNow[chInWord] = '\0';

words = increase(words, numOfWords);

words[numOfWords - 1] = new char[1024];

strcpy\_s(words[numOfWords - 1], sizeof(char) \* 100, wordNow);

strcpy\_s(wordNow, sizeof(char) \* 100, "");

chInWord = 0;

continue;

}

}

else

{

wordNow[chInWord] = text[i];

chInWord++;

}

i++;

if (text[i] == '\0')

if (chInWord != 0)

{

wordNow[chInWord] = '\0';

words = increase(words, numOfWords);

words[numOfWords - 1] = new char[1024];

strcpy\_s(words[numOfWords - 1], sizeof(char) \* 100, wordNow);

strcpy\_s(wordNow, sizeof(char) \* 100, "");

chInWord = 0;

continue;

}

}

return words;

}

void writeToFile(const char\* nameOfFile, const char\* content, int toTheEnd)

{

ofstream fout;

if (toTheEnd)

fout.open(nameOfFile, ios::out | ios::app);

else

fout.open(nameOfFile, ios::out);

if (!fout)

{

error("Cannot open the file!", 500);

return;

}

fout << content;

fout.close();

}

template<typename T>

T\* increase(T\* arr, int &size)

{

size++;

int sizeVar = size;

T\* arrRes = new T[sizeVar];

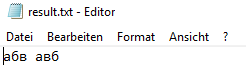
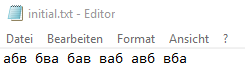
for (int i = 0; i < sizeVar - 1; i++)

arrRes[i] = arr[i];

return arrRes;

}

Результат выполнения:



**Задание 5 «Текстовые файлы с числовой информацией»**

В соответствии с индивидуальным заданием написать программу обработки содержимого файла.

Дана строка S, состоящая из 10 цифр, и файл с русским текстом. Зашифровать файл, выполнив циклическую замену каждой русской буквы, стоящей на K-й позиции строки, на букву того же регистра, расположенную в алфавите на SK-м месте после шифруемой буквы (для K = 11 снова используется смещение S1 и т. д.). Букву «ё» в алфавите не учитывать, знаки препинания и пробелы не изменять.

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <cmath>

#include <string>

#include <fstream>

#include <conio.h>

using namespace std;

void error(const char\*, int);

char \*\* readFromFile(char\*, int&);

int numOfLines(char\*);

template<typename T>

T\* increase(T\*, int&);

void writeToFile(const char\*, const char\*, int);

void encrypStr(char \*, char\*);

int isChInStr(char, char\* );

int main()

{

setlocale(LC\_ALL, "rus");

char nums[] = "0123456789";

int numOfLines;

char \*\* text = readFromFile("text.txt", numOfLines);

writeToFile("text.txt", "", 0);

for (int i = 0; i < numOfLines; i++)

{

encrypStr(text[i], nums);

writeToFile("text.txt", text[i], 1);

writeToFile("text.txt", "\n", 1);

}

system("pause");

return 0;

}

void encrypStr(char \* str, char\* nums)

{

char lowerCase[] = "абвгдежзийклмнопрстуфхцчшщъыьэюя";

char upperCase[] = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

int size = sizeof(lowerCase) / sizeof(\*lowerCase);

int j = 0;

for (int i = 0; str[i] != '\0'; i++)

{

int isLCase = isChInStr(str[i], lowerCase);

int isUCase = isChInStr(str[i], upperCase);

bool isInAlphabet = (isLCase + 1) || (isUCase + 1);

if (!isInAlphabet)

continue;

if (isLCase + 1)

{

str[i] = upperCase[(nums[(j % 10)] % size)];

j++;

}

if (isUCase + 1)

{

str[i] = lowerCase[(nums[(j % 10)] % size)];

j++;

}

}

}

int isChInStr(char ch, char\* str)

{

for (int i = 0; str[i] != '\0'; i++)

if (str[i] == ch)

return i;

return -1;

}

char \*\* readFromFile(char\* nameOfFile, int& numOfL)

{

ifstream fcin;

fcin.open(nameOfFile);

if (!fcin)

{

error("Cannot open the file!", 1000);

char \*\* error = new char\*;

return error;

}

numOfL = numOfLines(nameOfFile);

char \*\* lines = new char\*[numOfL];

for (int i = 0; i < numOfL; i++)

lines[i] = new char[100];

int i = 0;

string s;

for (; getline(fcin, s); i++)

strcpy\_s(lines[i], sizeof(s), s.c\_str());

return lines;

}

int numOfLines(char\* nameOfFile)

{

ifstream fcin(nameOfFile);

int numOfLines = 0;

string s;

while (getline(fcin, s))

numOfLines++;

return numOfLines;

}

void error(const char\* content, int timer)

{

system("cls");

cout << content << endl;

Sleep(timer);

system("cls");

}

void writeToFile(const char\* nameOfFile, const char\* content, int toTheEnd)

{

ofstream fout;

if (toTheEnd)

fout.open(nameOfFile, ios::out | ios::app);

else

fout.open(nameOfFile, ios::out);

if (!fout)

{

error("Cannot open the file!", 500);

return;

}

fout << content;

fout.close();

}

template<typename T>

T\* increase(T\* arr, int &size)

{

size++;

int sizeVar = size;

T\* arrRes = new T[sizeVar];

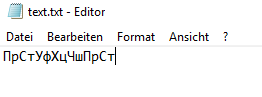
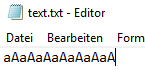
for (int i = 0; i < sizeVar - 1; i++)

arrRes[i] = arr[i];

return arrRes;

}

Результат выполнения:



Вывод: научился работать с текстовыми файлами.

Блок схема функции сортировки

