Балтийский государственный технический университет «Военмех»  
им. Д.Ф. Устинова

Кафедра И5

«Информационные системы и программная инженерия»

Индивидуальное домашнее задание №2  
дисциплина: «Основы программирования»  
тема: «Агрегатные структуры данных языка СИ»

Выполнил:

студент гр. ЗИ501 Житенев А.В.

Проверил:

Палехова О.А.

Санкт-Петербург  
2021 г.

***Вариант 8***

Задача 1. Определить, есть ли в массиве Q (10) заданное число Х, и если нет, то найти

ближайшее к нему.

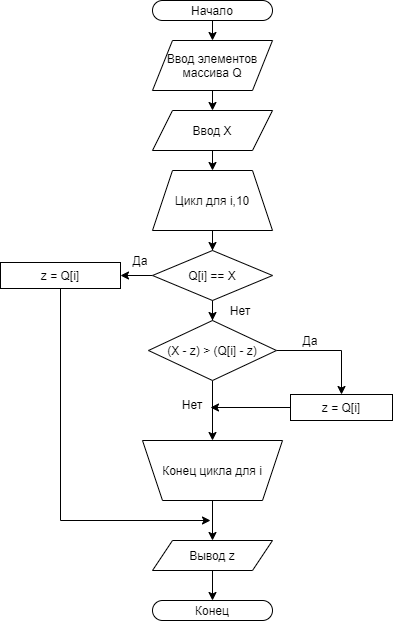
*Исходные данные:*

Целочисленный массив Q, искомое число X.

*Результирующие данные:*

Целочисленная переменная z.

*Схема программы*



*Текст программы*

#include <stdio.h>

void main(void) {

int Q[10];

int X;

int z = 0;

for (int i = 0; i<10; i++){

printf("Q[%d] = ", i+1);

scanf\_s("%d", &Q[i]);

}

printf("X = ");

scanf\_s("%d", &X);

for (int i = 0; i<10; i++) {

if (Q[i] == X) {

z = Q[i];

break;

}

if ((X - z) > (Q[i] - z)) {

z = Q[i];

}

}

printf("Z = %d", z);

return;

}

*Результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| Q=1,2,3,4,5,6,7,8,9,10 X=5 | 5 | Z = 5 |
| Q=1,2,3,4,6,7,8,9,10,11 X=5 | 4 | Z = 4 |

Задача 2. Поменять местами первый и последний отрицательные элементы массива В(18).

*Исходные данные:*

Целочисленный массив B,

*Вспомогательные переменные:*

Целочисленные переменные fNeg, lNeg для позиции первого и последнего отрицательного числа соответственно.

*Результирующие данные:*

Массив B, где первое и последнее отрицательные числа поменяны местами.

*Текст программы*

#include <stdio.h>

void main(void) {

int B[18];

int fNeg = -1, lNeg = -1;

for (int i = 0; i<18; i++){

printf("B[%d] = ", i+1);

scanf\_s("%d", &B[i]);

}

for (int i = 0; i<18; i++) {

if (B[i] < 0 && fNeg<0) fNeg = i;

if (B[i] < 0) lNeg = i;

}

int temp = B[fNeg];

B[fNeg] = B[lNeg];

B[lNeg] = temp;

for (int i = 0; i<5; i++) {

printf("%d,",B[i]);

}

return;

}

*Результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| B=1,-4,3,-2,5,6,7,8,9,10,11,12,13,  14,15,16,17,18 | B=1,-2,3,-4,5,6,7,8,9,10,  11,12,13,14,15,16,17,18 | B=1,-2,3,-4,5,6,7,8,9,10,  11,12,13,14,15,16,17,18 |
| B=-18,2,3,4,5,6,7,8,9,10,11,12,13,  14,15,16,17,-1 | B=-1,2,3,4,5,6,7,8,9,10,  11,12,13,14,15,16,17,-18 | B=-1,2,3,4,5,6,7,8,9,10,  11,12,13,14,15,16,17,-18 |

Задача 3. Удалить из массива М (26) первый положительный элемент.

*Исходные данные:*

Целочисленный массив M.

*Результирующие данные:*

Целочисленный массив A.

*Вспомогательные переменные:*

Целочисленная переменная k – позиция первого положительного числа массива M.

*Текст программы*

#include <stdio.h>

void main(void) {

const int N = 26, Q = 25;

int M[N], A[Q];

int k = -1;

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

printf("M[%d] = ", i+1);

scanf\_s("%d", &M[i]);

}

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

if (M[i] > 0) {

k = i;

break;

}

}

for (int j = 0, i = 0; j < Q; j++, i++){

if (i == k) i++;

A[j] = M[i];

}

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

printf("%d,", A[i]);

}

return;

}

Результаты тестирования

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| M = 1,2,3,4,5,..,25,26 | 2,3,4,5,…,25,26 | 2,3,4,5,…,25,26 |
| M = -2,-3,5,-6,…,-25,-26 | -2,-3,-6,…,-25,-26 | -2,-3,-6,…,-25,-26 |

Задача 4. Найти разность между суммами элементов, лежащих на главной и побочной

диагоналях матрицы М (7х7)

*Исходные данные:*

Количество строк и столбцов – N двумерного массива. Размер задано константой #define N 7.

Двумерный массив M типа int.

*Результирующие данные:*

Переменная diagA типа int для суммы элементов главной диагонали, переменная diagB типа int для суммы элементов побочной диагонали.

*Промежуточные данные:*

Целочисленные переменные i, j (int i, j;) – для организации циклов.

*Используемые алгоритмы обработки данных:*

1. Формирование массива генерацией псевдослучайных чисел.

2. Алгоритм отбора элементов, удовлетворяющих условию.

3. Алгоритм подсчета суммы.

*Схема программы:*



*Текст программы*

#include <stdio.h>

#include <stdlib.h>

#define N 7

void main(void) {

int M[N][N], diagA = 0, diagB = 0;

srand(42);

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

for (int j = 0; j < N; j++) {

M[i][j] = rand() % 10;

}

}

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

for (int j = 0; j < N; j++) {

printf("%i", M[i][j]);

}

putchar('\n');

}

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

diagA += M[i][i];

}

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

diagB += M[i][j];

}

printf("Основная диагональ: %i", diagA);

printf("Побочная диагональ: %i", diagB);

}

Результаты тестирования

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| 1740948  8245517  1152761  4232216  8576189  2795431  2334113 | 17, 34 | Основная диагональ: 17  Побочн6ая диагональ: 34 |
| 5096396  4999019  6582167  0670061  5527856  7927593  3809195 | 44, 22 | Основная диагональ: 44  Побочн6ая диагональ: 22 |

Задача 5. Отсортировать строки матрицы А(6х7) в порядке убывания.

*Исходные данные:*

Количество строк N и столбцов K двумерного массива. Максимальные размеры

зададим константами с помощью препроцессорной директивы

#define N 6

#define M 7

Двумерный массив M типа int.

*Результирующие данные:*

Переменная diagA типа int для суммы элементов главной диагонали, переменная diagB типа int для суммы элементов побочной диагонали.

*Промежуточные данные:*

Целочисленные переменные i, j, z – для организации циклов.

*Используемые алгоритмы обработки данных:*

1. Формирование массива генерацией псевдослучайных чисел.

2. Алгоритм отбора элементов, удовлетворяющих условию.

3. Алгоритм сортировки «пузырьком»

*Текст программы*

#include <stdio.h>

#include <stdlib.h>

#define N 6

#define K 7

void main(void) {

int M[N][K];

srand(45);

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

for (int j = 0; j < K; j++) {

M[i][j] = rand() % 10;

}

}

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

for (int j = 0; j < K; j++) {

printf("%i", M[i][j]);

}

putchar('\n');

}

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

for (int j = 0; j < K - 1; j++) {

for (int z = 0; z < K - j - 1; z++) {

if (M[i][z] < M[i][z + 1]) {

int tmp = M[i][z];

M[i][z] = M[i][z + 1];

M[i][z + 1] = tmp;

}

}

}

}

putchar('\n');

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

for (int j = 0; j < K; j++) {

printf("%i", M[i][j]);

}

putchar('\n');

}

}

Результаты тестирования

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| 5562903  0635320  6382346  7431860  6261578  4604815 | 9655320  6533200  8664332  8764310  8766521  8654410 | 9655320  6533200  8664332  8764310  8766521  8654410 |
| 9951768  6091425  8588233  3323847  8054166  3556208 | 9987651  9654210  8885332  8743332  8665410  8655320 | 9987651  9654210  8885332  8743332  8665410  8655320 |

Задача 6. Дана символьная строка. Зашифровать в ней все латинские буквы с помощью

циклической подстановки A->B->C->…->Y->Z->A

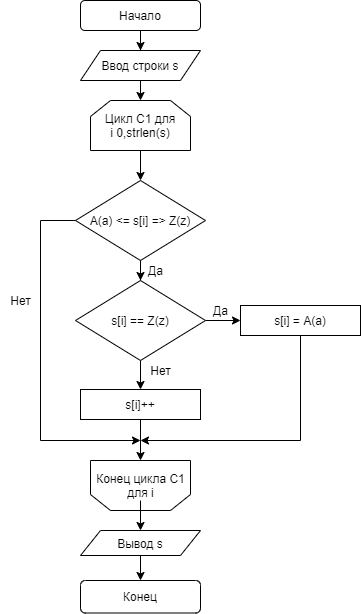
*Исходные данные:*

Символьный массив для хранения строки char s[64].

*Результирующие данные:*

Измененная строка s.

*Схема программы*

**

*Текст программы*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void main(void) {

char s[64];

printf("Type string: ");

gets\_s(s);

for (int i = 0; i < strlen(s); i++){

if (s[i] > 64 && s[i] < 91 || s[i]>96 && s[i]<123) {

if (s[i] == 90) s[i] = 65;

else if (s[i] == 122) s[i] = 97;

else s[i]++;

}

}

puts(s);

}

Результаты тестирования

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| Caeser | Dbftfs | Dbftfs |
| Sator Arepo Tenet | Tbups Bsfqp Ufofu | Tbups Bsfqp Ufofu |

Задача 7. Дана строка символов до точки. Определить, входят ли в состав заданной строки цифры. Сформировать из них новую строку.

*Исходные данные:*

Символьный массив для хранения строки char s[64].

*Результирующие данные:*

Символьный массив для хранения строки с цифрами char numb[64].

*Вспомогательные переменные:*

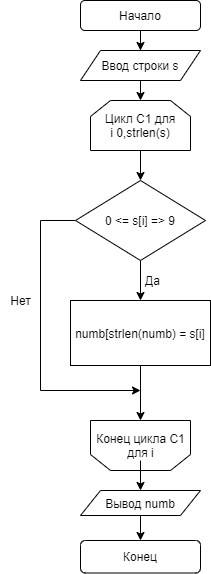
Целочисленные переменные i – для организации циклов для прохода

по строке.

*Используемые алгоритмы обработки данных:*

Алгоритм отбора элементов, удовлетворяющих условию.

*Схема программы*



*Текст программы*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void main(void) {

char s[64];

char numb[64] = {'\0'};

printf("Type string: ");

gets\_s(s);

for (int i = 0; i < strlen(s); i++){

if (s[i] > 47 && s[i] < 58) {

numb[strlen(numb)] = s[i];

}

}

puts(numb);

}

Результаты тестирования

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| N1UM2B3E4R. | 1234 | 1234 |
| NUMBER. |  |  |
| 1234. | 1234 | 1234 |

Задача 8. Сформировать массив для хранения информации о расписании движения

поездов: пункт назначения, номер поезда, тип поезда (скорый, экспресс, пассажирский), время отправления, время в пути. Вывести сведения о поездах,

отправляющихся в Москву в определенный временной период. Найти поезд

определенного типа, доезжающий до Москвы за наименьшее время.

*Исходные данные:*

1. Размер массива N – целочисленная константа (воспользуемся препроцессорной директивой #define N 5).

2. Структура train, включающая следующие поля: пункт назначения – char destination[64]; номер – char number[16]; тип – char type[64]; время отправления – int departure, время в пути - int travelTime.

3. Массив структур schedule заданного размера N: struct schedule train[N];

4. Переменные типа char from и till – временной промежуток.

5. Переменная char type – вид поезда, идущего в Москву за кратчайшее время.

*Результирующие данные:*

1. Массив структур, выведенные на экран в виде таблицы.
2. Структура, содержащая сведения о поездах, отправляемых в Москву в определенное время.
3. Структура, содержащая сведения о самом быстром поезде в Москву определенного типа.

*Вспомогательные переменные:*

1. Целочисленная переменная i (int i;) – для организации циклов.
2. Целочисленная переменная z (int z;) – для поиска самого быстрого поезда.

*Используемые алгоритмы обработки данных:*

1. Формирование структуры.
2. Формирование массива.
3. Сравнение строк.
4. Вывод массива на экран.

*Текст программы*

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#define N 5

struct schedule{

char destination[64];

char number[16];

char type[64];

int departure;

int travelTime;

};

int strToTime(char \*str) {

int time = 0;

for (int i = 0; i < strlen(str); i++){

if (str[i] != ':') time = (time \* 10) + (str[i]-'0');

}

return time;

}

void main(void){

struct schedule train[N];

char type[64];

char departure[10];

char travelTime[10];

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

puts("Number: ");

gets\_s(train[i].number);

puts("Destination:");

gets\_s(train[i].destination);

puts("Type: ");

gets\_s(train[i].type);

puts("Departure time: ");

gets\_s(departure);

train[i].departure = strToTime(departure);

puts("Travel time: ");

gets\_s(travelTime);

train[i].travelTime = strToTime(travelTime);

}

\_getch();

system("CLS");

puts("| N | Destination | Number | Type | Departure | Travel time");

puts("---------------------------------------------------------------------");

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

printf("|%3d|%13s|%8s|%15s|%11i|%12i|\n", i + 1, train[i].destination,

train[i].number, train[i].type, train[i].departure, train[i].travelTime);

}

\_getch();

char from[10], till[10];

puts("Train to Moscow from: ");

gets\_s(from);

puts("till: ");

gets\_s(till);

puts("| N | Destination | Number | Type | Departure | Travel time");

puts("---------------------------------------------------------------------");

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

if (!strcmp(train[i].destination, "Moscow") && train[i].departure >= strToTime(from) && train[i].departure < strToTime(till)) {

printf("|%3d|%13s|%8s|%15s|%11i|%12i|\n", i + 1, train[i].destination,

train[i].number, train[i].type, train[i].departure, train[i].travelTime);

}

}

int z = 0;

puts("Fastet train to Moscow: ");

puts("Type of train: ");

gets\_s(type);

for (int i = 1; i < N; i++) {

if (!strcmp(train[i].destination, "Moscow") && !strcmp(train[i].type, type) && train[i].travelTime < train[z].travelTime) z = i;

}

puts("| N | Destination | Number | Type | Departure | Travel time");

puts("---------------------------------------------------------------------");

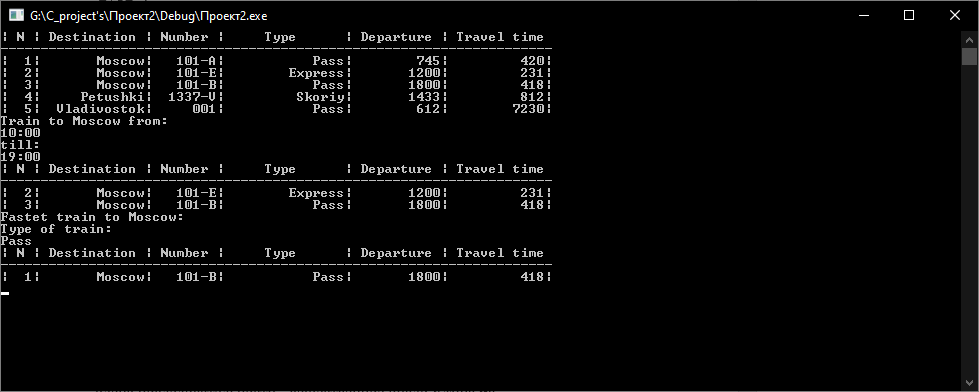
printf("|%3d|%13s|%8s|%15s|%11i|%12i|\n", 1, train[z].destination,

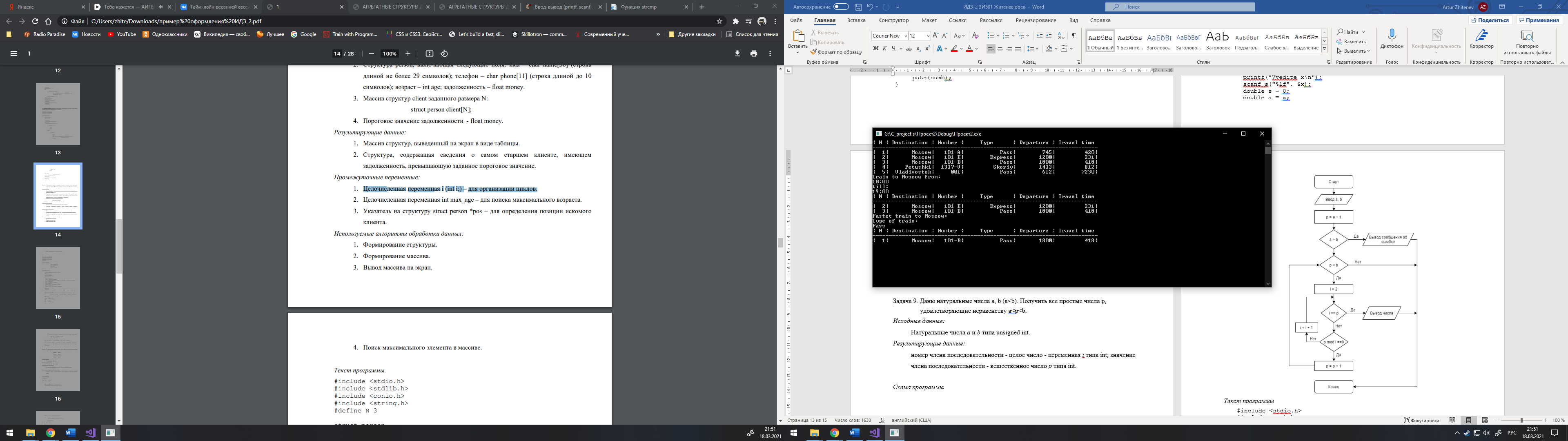
train[z].number, train[z].type, train[z].departure, train[z].travelTime);

\_getch();

return;

}

Результаты тестирования 



Задача 9. Дан текстовый файл, в каждой строке которого записано по несколько слов,

разделенных пробелом. Создать новый текстовый файл, в каждой строке

которого записаны слова, начинающиеся с буквы ‘a’ из соответствующей

строки исходного файла.

*Исходные данные:*

1. Текстовый файл input.txt с различными словами.
2. Переменные типа char[128] string, word для считывания текста из файла и отделения отдельных слов соответственно. Указатель char \*lexeme – указатель на отдельную лексему в файле.

*Используемые алгоритмы обработки данных:*

1. Алгоритм чтения из файла данных с помощью функции fgets().
2. Алгоритм записи данных файл с помощью функции fprintf().
3. Алгоритм поиска лексем в строке strtok().
4. Алгоритм копирования строк strcopy().

*Результирующие данные:*

Текстовый файл output.txt со словами из input.txt, который начинаются с “a”.

*Текст программы*

#include <stdio.h>

#include <string.h>

void main(){

FILE \*fin, \*fout;

fin = fopen("input.txt", "r");

fout = fopen("output.txt", "w");

char string[128], word[128], \*lexeme;

while (fgets(string, 128, fin)) {

for (lexeme = strtok(string, " \n"); lexeme; lexeme = strtok(NULL, " \n")) {

strcpy(word, lexeme);

if (word[0] == 'a' || word[0] == 'A') fprintf(fout, "%s ", word);

}

fprintf(fout, "\n");

}

fclose(fin);

fclose(fout);

return;

}

Задача 10. Дан текстовый файл, который содержит целые числа по несколько в каждой

строке. Найти сумму первой и последней компонент в каждой строке файла и

записать найденные суммы в новый текстовый файл.

*Исходные данные:*

1. Текстовый файл input.txt с числами.
2. Переменные типа int number = считанное число, x – первое число строки, y – последнее число строки.
3. Переменная типа char ch – символ идущий после числа.
4. Переменная типа bool – указатель наличия первого числа в строке.

*Используемые алгоритмы обработки данных:*

1. Алгоритм чтения из файла данных с помощью функции fscanf().
2. Алгоритм записи данных файл с помощью функции fprintf().

*Результирующие данные:*

Текстовый файл output.txt с суммами первых и последних чисел в строке.

*Текст программы*

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

void main(){

FILE\* input = fopen("input.txt", "r");

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

int number, x, y;

char ch;

bool i = 0;

while(1){

fscanf(input, "%d%c", &number, &ch);

printf("%d%c", number, ch);

if (!i) {

x = number;

i = true;

}

if (ch == '\n' || feof(input)) {

y = number;

fprintf(output, "%d\n", x + y, "\n");

i = false;

}

if (feof(input)) break;

}

fclose(input);

fclose(output);

puts("\n");

system("pause");

return;

}

Задача 11. В текстовом файле построчно хранится матрица размером не больше 5х4.

Записать в другой файл количество положительных, отрицательных и нулевых

элементов исходной матрицы, ее среднее арифметическое значение, максимум

и минимум (с позициями).

*Исходные данные:*

1. Текстовый файл input.txt с матрицей 5х4.
2. Двумерный массива int array[N][M] для массива, переменные int positive = 0, negative = 0, null = 0, total = 0 для необходимых данных, переменная double average для среднего арифметического матрицы.
3. Массивы max[3] = { 0,0,0 }, min[3] = { 0,0,0 } для максимального и минимального числа в массиве и их позиция.

*Используемые алгоритмы обработки данных:*

1. Алгоритм чтения из файла данных с помощью функции fscanf().
2. Алгоритм записи данных файл с помощью функции fprintf().

*Результирующие данные:*

Текстовый файл output.txt в котором: количество положительных, отрицательных и нулевых элементов исходной матрицы, ее среднее арифметическое значение, максимум и минимум (с позициями).

*Текст программы*

#include<stdio.h>

#define N 5

#define M 4

void main(){

FILE \*input, \*output;

int array[N][M];

int positive = 0, negative = 0, null = 0, total = 0;

int max[3] = { 0,0,0 }, min[3] = { 0,0,0 };

double average = 0;

if ((input = fopen("input.txt", "r")) == NULL){

printf("\nREAD ERROR!\n");

return;

}

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

for (int j = 0; j<4; j++){

fscanf(input, "%d", &array[i][j]);

printf("%d ", array[i][j]);

if (array[i][j] > 0) positive++;

else if (array[i][j] < 0) negative++;

else if (array[i][j] == 0) null++;

if (array[i][j] > max[0]) {

max[0] = array[i][j];

max[1] = i; max[2] = j;

}

if (array[i][j] < min[0]) {

min[0] = array[i][j];

min[1] = i; min[2] = j;

}

total++;

average += array[i][j];

}

printf("\n");

}

average = average / total;

output = fopen("output.txt", "w");

fprintf(output, "Positive = %d \nNegative = %d \nNull = %d \nAverage = %f \nMaximum = %d[%d][%d] \nMinimum = %d[%d][%d]", positive, negative, null, average, max[0], max[1], max[2], min[0], min[1], min[2]);

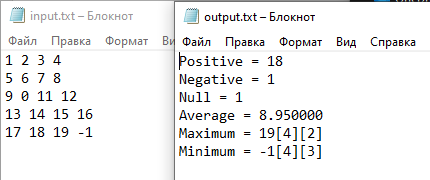
fclose(input);

fclose(output);

return;

}

*Результаты тестирования:*



Задача 12. Компоненты бинарного файла – вещественные числа. Изменить содержимое

файла, прибавив к каждой положительной компоненте первую, а из

отрицательных компонент вычесть значение последней.

*Исходные данные:*

1. Последовательность вещественных чисел – double numbers[].

*Используемые алгоритмы обработки данных:*

1. Алгоритм чтения из файла данных с помощью функции fread().
2. Алгоритм записи данных файл с помощью функции fwrite().
3. Алгоритм динамического выделения памяти malloc().

*Результирующие данные:*

Бинарный файл, с выполненным условием. Вывод измененных, в соответствии с условием, вещественных чисел.

*Текст программы*

#include <stdio.h>

#include <malloc.h>

void create\_binary();

void read\_binary();

void change\_binary();

void main(){

create\_binary();

read\_binary();

change\_binary();

}

void create\_binary() {

FILE\* f = fopen("binary", "wb");

const int K = 5;

double numbers[K] = { 3.5, 4.8, -2.56, -1.33, 2.18 };

fwrite(&K, sizeof(int), 1, f);

for (int i = 0; i < 5; i++) {

fwrite(&numbers[i], sizeof(double), 1, f);

printf("%f\n", numbers[i]);

}

puts("Data writeen sucessfully...\n");

fclose(f);

}

void read\_binary() {

FILE\* f = fopen("binary", "rb");

if (f == NULL) {

printf("Error opening file");

return;

}

int K = 0;

fread(&K, sizeof(int), 1, f);

double \*numbers = (double\*)malloc(sizeof(double) \* K);

if (numbers == NULL) return;

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

fread(&numbers[i], sizeof(double), 1, f);

}

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

printf("%f\n", numbers[i]);

}

puts("Data readed sucessfully...\n");

fclose(f);

}

void change\_binary() {

FILE\* f = fopen("binary", "r+b");

if (f == NULL) {

printf("Error opening file");

return;

}

int K = 0;

fread(&K, sizeof(int), 1, f);

double \*numbers = (double\*)malloc(sizeof(double) \* K);

if (numbers == NULL) return;

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

fread(&numbers[i], sizeof(double), 1, f);

}

double F = numbers[0]; double L = numbers[K - 1];

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

if (numbers[i] > 0) numbers[i] += F;

else if (numbers[i] < 0) numbers[i] -= L;

}

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

printf("%f\n", numbers[i]);

}

fclose(f);

}

Задача 13. Компоненты бинарного файла – массивы, состоящие из 5 целых чисел. Вывести

на экран те из них, которые не содержат четных значений.

*Исходные данные:*

1. 3 массива из 5 натуральных чисел – int numbers[3][5].

*Используемые алгоритмы обработки данных:*

1. Алгоритм чтения из файла данных с помощью функции fread().
2. Алгоритм записи данных файл с помощью функции fwrite().
3. Алгоритм динамического выделения памяти malloc().

*Результирующие данные:*

Текстовый файл output.txt со словами из input.txt, который начинаются с “a”.

*Текст программы*

#include <stdio.h>

#include <malloc.h>

void create\_binary();

void read\_binary();

void change\_binary();

void main(){

create\_binary();

read\_binary();

change\_binary();

}

void create\_binary() {

FILE\* f = fopen("binary", "wb");

const int K = 3, M = 5;

int numbers[K][M] = { 1, 1, 1, 1, 5, 6, 7, 8, 9, 10, 11, 19, 13, 13, 15 };

fwrite(&K, sizeof(int), 1, f);

fwrite(&M, sizeof(int), 1, f);

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

for (int j = 0; j < M; j++){

fwrite(&numbers[i][j], sizeof(int), 1, f);

}

}

puts("Data written sucessfully...\n");

fclose(f);

}

void read\_binary() {

FILE\* f = fopen("binary", "rb");

if (f == NULL) {

printf("Error opening file");

return;

}

int K = 0, M = 0;

fread(&K, sizeof(int), 1, f);

fread(&M, sizeof(int), 1, f);

int\*\* numbers = (int\*\*)malloc(K\* sizeof(int\*));

for (int i = 0; i < K; ++i) numbers[i] = (int\*)malloc(M \* sizeof(int));

if (numbers == NULL) return;

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

for (int j = 0; j < M; j++) {

fread(&numbers[i][j], sizeof(int), 1, f);

}

}

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

for (int j = 0; j < M; j++){

printf("%3d", numbers[i][j]);

}

puts("\n");

}

puts("Data readed sucessfully...\n");

fclose(f);

}

void change\_binary() {

FILE\* f = fopen("binary", "r+b");

if (f == NULL) {

printf("Error opening file");

return;

}

int K = 0, M = 0;

fread(&K, sizeof(int), 1, f);

fread(&M, sizeof(int), 1, f);

int\*\* numbers = (int\*\*)malloc(K \* sizeof(int\*));

for (int i = 0; i < K; ++i) numbers[i] = (int\*)malloc(M \* sizeof(int));

if (numbers == NULL) return;

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

for (int j = 0; j < M; j++) {

fread(&numbers[i][j], sizeof(int), 1, f);

}

}

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

bool q = 0;

for (int j = 0; j < M; j++) {

if (!(numbers[i][j] % 2)) q = true;

}

if (!q) {

for (int j = 0; j < M; j++) {

printf("%3d", numbers[i][j]);

}

puts("\n");

}

}

fclose(f);

}

*Результаты тестирования:*