# Tasks:

# Билет 1.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 1. Дано 2 бинарных файла. Один из них отсортирован по возрастанию, второй - по убыванию. Записать всё в третий файл по убыванию без дополнительных массивов.

## 2. Проход по дереву рекурсивно, вывод по возрастанию.

# Билет 2

## 1. Дано 2 текстовых файла, отсортированных по возрастанию. Сделать один, отсортированный по убыванию.

## 2. Дано бинарное дерево. Организовать удаление любого узла (элемента).

# Билет 3

## 1. Отсортировать текстовый файл методом отбора.

## 2. Осуществить нерекурсивный проход по бинарному дереву, чтобы печатался сначала предок, а потом потомок.

# Билет 4

## 1. Отсортировать бинарный файл методом вставок.

## 2. Удалить из двунаправленного кольца любой элемент.

# Билет 5

## 1. Дан бинарный файл, отсортированный по возрастанию. Ввести числа в файл, не нарушая очередности.

## 2. Организовать нерекурсивный просмотр дерева, чтобы вначале выводился потомок, потом предок.

# Билет 6

## 1. В командной строке задаётся имя файла и диапазон чисел. Удалить из бинарного файла числа из данного диапазона.

## 2. Сортировка однонаправленной очереди. Элемент очереди – указатель на символьную строку.

# Билет 7

## 1. В файле записаны структуры. Фамилия и 3 оценки (тип файла выбрать самостоятельно). Удалить все записи с минимальной суммой оценок. Имя файла задаётся в командной строке.

## 2. Написать функцию сортировки двунаправленного кольца заменой указателей.

# Билет 8

## 1. Закодировать текстовый файл без использования дополнительных массивов (т.е. поменять местами первое и последнее слово, второе и предпоследнее и т.д.).

## 1. Закодировать текстовый файл без использования дополнительных массивов (т.е. поменять местами первый и последний символы, второй и предпоследнийи т.д.).

## 2. Удалить любой элемент из однонаправленного кольца.

# Билет 9

## 1. В бинарном файле поменять местами 1 и 2, 3 и 4 элементы и т.д.

## 2. В структуре храниться имя фигуры, в ней (в структуре) объединение, в котором хранятся параметры фигуры (радиус окружности, периметрпрямоугольника). Запросить из командной строки название 10 фигур и подсчитать их площадь по отдельности.

# Билет 10

## 1. Отсортировать бинарный файл целых длинных чисел методом вставок.

## 2. Удалить любой элемент из однонаправленной очереди.

# Билет 11

## 1. Даны 2 бинарных файла, отсортированных по убыванию. Сделать один, отсортированный по возрастанию.

## 2. Дано однонаправленное кольцо с указателями на стек и на строку. Найти элемент с максимальной длиной строки, удалить его, а стек переписать в следующий элемент кольца.

# Билет 12

## 1. Даны текстовый и бинарный файлы. Диапазон чисел в этих файлах от 1 до 999. Из текстового файла удалить все числа, встречающиеся в бинарном.

## 2. Функция нерекурсивного создания бинарного дерева.

# Билет 13

## 1. Даны текстовый и бинарный файлы. Из бинарного файла удалить все числа, встречающиеся в текстовом.

## 2. Элемент однонаправленного кольца – структура. Отсортировать кольцо методом вставок по названию структур.

# Билет 14

## 1. Есть бинарный файл, упорядоченный по возрастанию. Вставить в него целое число, не используя дополнительных массивов сортировки, чтобы не нарушать последовательность.

## 2. Из текстового файла добавлять и упорядочивать слова в дерево.

# Билет 15

## 1. Дан текстовый файл, содержащий двузначные числа. Поменять местами первое и последнее число, второе и предпоследнее и т.д.

## 2. Написать функцию добавления элемента в бинарное дерево рекурсивно (элементы: фамилия, год).

# Билет 16

## 1. Используя поля бит, найти остаток от деления на 2 и 4.

## 2. Бинарное дерево записать в файл, имя которого вводится с командной строки.

# Билет 17

## 1. Дано 2 бинарных файла, отсортированных по возрастанию и убыванию. Создать третий, отсортированный по возрастанию.

## 2. Рекурсивно создать двунаправленное кольцо.

# Билет 18

## 1. Записать в битовом представлении буквы алфавита.

## 2. Дана однонаправленная очередь, упорядоченная по убыванию. Записать в бинарный файл, имя которого задаётся в командной строке, содержимое

очереди по возрастанию без дополнительных массивов и файлов.

# Билет 19

## 1. Удаление элемента из однонаправленной очереди.

## 2. Разработать программу «Библиотека». Структура: автор, название, год. Должны работать функции добавления и сортировки.

# Билет 20

## 1. Удалить из текстового файла повторяющиеся символы.

## 2. Организовать ввод информации в массив, организованный по принципу бинарного дерева (для k-ого элемента: если левое поддерево, то сделать 2k, если правое, то 2k+1).

# Билет 21

## 1. Даны два файла, упорядоченные по возрастанию. Создать один, отсортированный по убыванию. Все файлы бинарные.

## 2. Даны массивы структур из двух элементов, содержащих указатели на структуры. 1-я структура – имя студента и группа, вторая – 4 балла. Найти студента с наибольшим баллом.

//bilet 1 Проход по дереву рекурсивно, вывод по возрастанию.

typedef struct TreeNode {

void\* data;

TreeNode\* left;

TreeNode\* right;

}TreeNode

void treeTravel(TreeNode\* root, void (\*function)(void\*)) {

if (root) {

treeTravel(root->left, function);

printData(root->data);

treeTravel(root->right, function);

}

}

void printData(void\* data) {

printf("%d ", \*((int\*)data));

}

//bilet 2 Дано бинарное дерево. Организовать удаление любого узла(элемента)

typedef struct TreeNode{

int data;

TreeNode\* left;

TreeNode\* right;

}TreeNode;

TreeNode\* getMaxNode(TreeNode\* root) {

while (root->right) {

root = root->right;

}

return root;

}

TreeNode\* deleteNode(TreeNode\* root, int value) {

if (root == NULL) {

return root;

}

// поиск данных по значению

if (root->data > value) {

root->left = deleteNode(root->left, value);

return root;

}

else if (root->data < value) {

root->right = deleteNode(root->right, value);

return root;

}

else {

if (root->right && root->left) {

TreeNode\* max = getMaxNode(root->left);

root->data = max->data;

root->left = deleteNode(root->left, max->data);

return root;

}

else if (root->left) {

TreeNode\* tmp = root->left;

free(root);

return tmp;

}

else if (root->right) {

TreeNode\* tmp = root->right;

free(root);

return tmp;

}

else {

free(root);

return NULL;

}

}

}

//bilet 3 Отсортировать текстовый файл методом отбора

void enterFile(FILE\* file, const char\* filename, int\* arr, int size) {

file = fopen(filename, "w");

if (file == NULL) {

printf("Error\n");

exit(-1);

}

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

fprintf(file, "%5d", arr[i]);

}

fclose(file);

}

int\* Arr(int\* size) {

int\* array = (int\*)malloc(\*size \* sizeof(int));

if (array == NULL) {

exit(-2);

}

return array;

}

int filesize(FILE\* file) {

fseek(file, 0, SEEK\_END);

return ftell(file);

}

int cmp\_gt(int a, int b) {

return a > b;

}

int cmp\_less(int a, int b) {

return a < b;

}

void fileSorting\_selective\_(FILE\* file, const char\* filename, int (\*cmp)(int, int)) {

file = fopen(filename, "r+");

if (file == NULL){

printf("Error opening file!\n");

exit(-1);

}

int buffer\_to\_compare;

int current\_buffer;

int a, b;

for (fpos\_t i = 0; i < filesize(file); i += 5) {

fsetpos(file, &i);

fpos\_t \_position = ftell(file);

for (fpos\_t j = i; j < filesize(file); j += 5) {

fsetpos(file, &\_position);

fscanf(file, "%5d", &buffer\_to\_compare);

fsetpos(file, &j);

fscanf(file, "%5d", &current\_buffer);

if (cmp(buffer\_to\_compare, current\_buffer)) {

\_position = j;

}

}

fsetpos(file, &i);

fsetpos(file, &\_position);

fscanf(file, "%5d", &b);

fsetpos(file, &i);

fprintf(file, "%5d", b);

fsetpos(file, &\_position);

fprintf(file, "%5d", a);

}

fclose(file);

}

int main() {

FILE\* file;

int size = 0;

int\* arr = Arr(&size);

enterFile(file, "f.txt", arr, size);

fileSorting\_selective\_(file, "f.txt", cmp\_gt);

fileSorting\_selective\_(file, "f.txt", cmp\_less);

free(arr);

}

//bilet 4 Отсортировать бинарный файл методом отбора

int\* Arr(int\* size) {

int\* array = (int\*)malloc(\*size \* sizeof(int));

if (!arr) {

exit(error);

}

return array;

}

int filesize(FILE\* file) {

fseek(file, 0, SEEK\_END);

return ftell(file);

}

int cmp\_gt(int a, int b) {

return a > b;

}

int cmp\_less(int a, int b) {

return a < b;

}

void enterFile(FILE\* file, const char\* filename, int\* data, int size) {

file = fopen(filename, "wb");

if (file == NULL) {

printf("Error opening file!\n");

exit(-101);

}

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

fwrite(&data[i], sizeof(int), 1, file);

fclose(file);

}

int getFileData(FILE\* file, fpos\_t index) {

int data;

fsetpos(file, &index);

fread(&data, sizeof(int), 1, file);

return data;

}

void swapValuesInFile(FILE\* file, fpos\_t index1, fpos\_t index2) {

int a, b;

fsetpos(file, &index1);

fread(&a, sizeof(int), 1, file);

fsetpos(file, &index2);

fread(&b, sizeof(int), 1, file);

fsetpos(file, &index1);

fwrite(&b, sizeof(int), 1, file);

fsetpos(file, &index2);

fwrite(&a, sizeof(int), 1, file);

}

void binaryFIle\_sort\_selective(FILE\* file, const char\* filename, int (\*cmp)(int, int)) {

file = fopen(filename, "r+b");

if (file == NULL) {

printf("Error opening file!\n");

exit(-101);

}

int buffer\_to\_compare;

int current\_buffer;

int a, b;

for (fpos\_t i = 0; i < filesize(file); i += sizeof(int)) {

fsetpos(file, &i);

fpos\_t \_position = ftell(file);

for (fpos\_t j = i; j < filesize(file); j += sizeof(int)) {

buffer\_to\_compare = getFileData(file, \_position);

current\_buffer = getFileData(file, j);

if (cmp(buffer\_to\_compare, current\_buffer)) {

\_position = j;

}

}

a = getFileData(file, i);

b = getFileData(file, \_position);

swapValuesInFile(file, i, \_position);

}

fclose(file);

}

int main() {

FILE\* file = NULL;

int size = 0;

int\* array = Arr(&size);

enterFile(file, "f.bin", array, size);

binaryFIle\_sort\_selective(file, "f.bin", cmp\_less);

binaryFIle\_sort\_selective(file, "f.bin", cmp\_gt);

free(array);

}

//bilet 5 вставить число в отсортированный по возрастанию бинарный файл не нарушая последовательности

int\* Arr(size\_t \_сreatedArrayLenght) {

int\* arr = (int\*)malloc(\_сreatedArrayLenght \* sizeof(int));

if (!arr) {

exit(error);

}

return arr;

}

int compare\_less(int \_firstValue, int \_secondValue){

return \_firstValue < \_secondValue;

}

int compare\_greater(int \_firstValue, int \_secondValue) {

return \_firstValue > \_secondValue;

}

void sortArray(int\* array, int \_valuesNumber, int (\*\_functionToCompare)(int, int)) {

for (size\_t i = 0; i < \_valuesNumber - 1; i++){

for (size\_t j = 0; j < \_valuesNumber - i - 1; j++){

if (\_functionToCompare(array[j], array[j + 1])) {

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

void enterFile(const char\* filename, int \_valuesNumber) {

FILE\* filePointer = fopen(filename, "wb");

if (filePointer == NULL) {

printf("Error opening file!\n");

exit(-101);

}

int\* arr = Arr(\_valuesNumber);

sortArray(arr, \_valuesNumber, compare\_greater);

for (size\_t i = 0; i < \_valuesNumber; i++){

fwrite(&arr[i], sizeof(int), 1, filePointer);

}

free(arr);

fclose(filePointer);

}

int filesize(FILE\* filePointer) {

fseek(filePointer, 0, SEEK\_END);

return ftell(filePointer);

}

int getFileData(FILE\* file, fpos\_t \_positionToRead) {

int buffer = 0;

fread(&buffer, sizeof(int), 1, file);

return buffer;

}

void insertValue(const char\* filename, int \_valueToInsert) {

FILE\* filePointer = fopen(filename, "rb+");

if (!filePointer) {

printf("Error openig file!\n");

exit(-101);

}

int currentBuffer;

int insertionMarker = 0;

int tempValue = 0;

int filelenght = filesize(filePointer);

for (fpos\_t i = 0; i < filelenght; i += sizeof(int)) {

fsetpos(filePointer, &i);

fread(&currentBuffer, sizeof(int), 1, filePointer);

if (currentBuffer >= \_valueToInsert) {

insertionMarker = 1;

for (fpos\_t j = i; j < filelenght + sizeof(int); j += sizeof(int)) {

fsetpos(filePointer, &j);

fread(&tempValue, sizeof(int), 1, filePointer);

fsetpos(filePointer, &j);

fwrite(&\_valueToInsert, sizeof(int), 1, filePointer);

\_valueToInsert = tempValue;

}

break;

}

}

if (!insertionMarker) {

fwrite(&\_valueToInsert, sizeof(int), 1, filePointer);

}

fclose(filePointer);

}

int main() {

int fileLenght = 10;

enterFile("a.txt", 10);

insertValue("a.txt", 7);

insertValue("a.txt", 3);

insertValue("a.txt", 6);

}

//bilet 6 В командной строке задаётся имя файла и диапазон чисел. Удалить из бинарного файла числа из данного диапазона

int\* Arr(int numval) {

int\* arr = (int\*)malloc(numval \* sizeof(int));

if (!arr) {

exit(error);

}

return arr;

}

FILE\* createbin(const char\* filename, int numval, int maxval) {

FILE\* bfp = fopen(filename, "wb+");

if (!bfp) {

exit(error);

}

fclose(bfp);

}

FILE\* delete(const char\* filename, int min, int max) {

FILE\* bfp = fopen(filename, "rb");

if (!bfp) {

exit(error);

}

if (min > max) {

int temp = min;

min = max;

max = temp;

}

int\* file\_data = (int\*)malloc(sizeof(int));

if (!file\_data) {

exit(error);

}

int buf = 0;

int i = 0;

int array\_size = 0;

while (!feof(bfp)) {

fread(&buf, sizeof(int), 1, bfp);

if (buf >= min && buf <= max) {

continue;

} else {

file\_data = (int\*)realloc(file\_data, (i + 1) \* sizeof(int));

array\_size++;

if (!file\_data) {

exit(error);

}

file\_data[i++] = buf;

}

}

bfp = fopen(filename, "wb");

if (!bfp) {

exit(error);

}

for (int i = 0; i < array\_size; i++) {

fwrite(&file\_data[i], sizeof(int), 1, bfp);

}

fclose(bfp);

return bfp;

}

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

const char\* filename = argv[0];

int min = atoi(argv[1]);

int max = atoi(argv[2]);

FILE\* bfp = createbin(filename, 10, 20);

bfp = delete(filename, min, max);

}

//bilet 7 сортировка двунаправленного кольца методом замены указателей

typedef struct Node {

int data;

Node\* next;

Node\* prev;

}Node;

typedef struct Ring {

Node\* current;

}Ring;

Ring\* createRing() {

Ring\* ring = (Ring\*)malloc(sizeof(Ring));

if (!ring) {

exit(error);

}

ring->current = NULL;

return ring;

}

int compare\_greater(int first\_value, int second\_value) {

return first\_value > second\_value;

}

int compare\_less(int first\_value, int second\_value) {

return first\_value < second\_value;

}

Ring\* sort(Ring\* ring, int (\*comparison\_function)(int, int)) {

Node\* main\_pointer = NULL,

\* current\_pointer = NULL,

\* pointer\_to\_swap = NULL;

main\_pointer = ring->current;

do

{

current\_pointer = main\_pointer->next;

pointer\_to\_swap = main\_pointer;

do { // проходим по списку

if (comparison\_function(pointer\_to\_swap->data, current\_pointer->data)) {

pointer\_to\_swap = current\_pointer;

}

current\_pointer = current\_pointer->next;

} while (current\_pointer != ring->current);

if (pointer\_to\_swap != main\_pointer) {

if (ring->current == main\_pointer) {

ring->current = pointer\_to\_swap;

}

// и делаем свап указателей

pointer\_to\_swap->prev->next = pointer\_to\_swap->next;

pointer\_to\_swap->next->prev = pointer\_to\_swap->prev;

main\_pointer->prev->next = pointer\_to\_swap;

pointer\_to\_swap->next = main\_pointer;

pointer\_to\_swap->prev = main\_pointer->prev;

main\_pointer->prev = pointer\_to\_swap;

}

else {

main\_pointer = main\_pointer->next;

}

} while (main\_pointer->next != ring->current);

return ring;

}

int main() {

Ring\* ring = createRing();

ring = sort(ring, compare\_greater);

ring = sort(ring, compare\_less);

}

//bilet 8 сделать реверс файла

void enterFile(const char\* filename, char\* data) {

FILE\* file = fopen(filename, "w");

if (file == NULL) {

exit(-1);

}

fclose(file);

}

int filesize(FILE\* file) {

fseek(file, 0, SEEK\_END);

return ftell(file);

}

char filedata(FILE\* file, fpos\_t position) {

char c = 0;

fsetpos(file, &position);

fscanf(file, "%c", &c);

return c;

}

void fileReverse(const char\* filename) {

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

if (file == NULL) {

exit(-101);

}

char c = 0;

int size = filesize(file);

for (fpos\_t i = 0; i < size / 2; i++) {

fpos\_t swap\_position = size - i - 1;

char temp = filedata(file, swap\_position);

fsetpos(file, &i);

fscanf(file, "%c", &c);

fsetpos(file, &swap\_position);

fsetpos(file, &i);

}

fclose(file);

}

int main() {

char data[40];

gets\_s(data);

enterFile("a.txt", data);

fileReverse("a.txt");

}

// Bilet 9 В бинарном файле поменять местами 1 и 2, 3 и 4

void swapfunc(FILE\* bin) {

int swap1[1], swap2[1];

fpos\_t pos1 = 0;

rewind(bin);

do {

fgetpos(bin, &pos1);

fsetpos(bin, &pos1);

fread(&swap1[0], sizeof(int), 1, bin);

if (feof(bin)) break;

fread(&swap2[0], sizeof(int), 1, bin);

if (feof(bin)) break;

fsetpos(bin, &pos1);

fwrite(&swap2[0], sizeof(int), 1, bin);

fwrite(&swap1[0], sizeof(int), 1, bin);

} while (1);

}

int main() {

FILE\* bin;

fopen\_s(&bin, my\_bin.bin, "w+");

if (!bin)

{

puts("Cant open file");

return 1;

}

swapfunc(bin);

return 0;

}

// bilet 10 Отсортировать бинарный файл целых длинных чисел методом отбора

void sort(FILE\* bin) {

int swap[2];

fpos\_t pos, pos\_end;

rewind(bin);

while (1){

fread(&swap[0], sizeof(int), 1, bin);

if (feof(bin)) break;

}

fgetpos(bin, &pos\_end);

rewind(bin);

pos = pos\_end - sizeof(int)\*2;

fsetpos(bin, &pos);

while (pos >= 0){

fread(&swap[0], sizeof(int), 1, bin);

fread(&swap[1], sizeof(int), 1, bin);

while (swap[1] < swap[0] && pos < pos\_end) {

fsetpos(bin, &pos);

fwrite(&swap[1], sizeof(int), 1, bin);

fpos\_t temp; fgetpos(bin, &temp);

fwrite(&swap[0], sizeof(int), 1, bin);

fsetpos(bin, &temp);

fread(&swap[0], sizeof(int), 1, bin);

fread(&swap[1], sizeof(int), 1, bin);

if (feof(bin))

break;

pos += sizeof(int);

}

pos -= sizeof(int); //спускаемся к началу

fsetpos(bin, &pos); //устанавливаем позицию

}

}

int main() {

FILE\* bin;

fopen\_s(&bin, my\_bin.bin, "w+");

if (!bin)

{

puts("error");

return 1;

}

sort(bin);

return 0;

}

//bilet 11 Даны 2 бинарных файла, отсортированных по убыванию. Сделать один

void sortfunc(FILE\* first, FILE\* second, FILE\* third) {

fpos\_t end1 = 0, end2 = 0;

int temp;

while (!feof(first))

fread(&temp, sizeof(int), 1, first);

fgetpos(first, &end1);

while (!feof(second))

fread(&temp, sizeof(int), 1, second);

fgetpos(second, &end2);

fpos\_t pos1 = end1 - sizeof(int), pos2 = end2 - sizeof(int);

// возможно у тебя вопрос: зачем делать -sizeof(int)?

// ответ прост - мы стоим на eof поэтому мы двигуаем на одну позицию назад

int select[2];

bool skip1 = false, skip2 = false;

rewind(first); rewind(second);

fsetpos(first, &pos1); fsetpos(second, &pos2);

while (pos1 >= 0 || pos2 >= 0) {

fsetpos(first, &pos1); fsetpos(second, &pos2);

if (pos1 >= 0)

fread(&select[0], sizeof(int), 1, first);

else

skip1 = 1;

if (pos2 >= 0)

fread(&select[1], sizeof(int), 1, second);

else

skip2 = 1;

if (skip1 == 1) {

fwrite(&select[1], sizeof(int), 1, third);

pos2 -= sizeof(int);

continue;

}

else (skip2 == 1) {

fwrite(&select[0], sizeof(int), 1, third);

pos1 -= sizeof(int);

continue;

}

if (select[0] < select[1]) {

fwrite(&select[0], sizeof(int), 1, third);

pos1 -= sizeof(int);

}

else {

fwrite(&select[1], sizeof(int), 1, third);

pos2 -= sizeof(int);

}

}

}

int main() {

FILE\* first, \* second, \* third;

fopen\_s(&first\second\third, first\second\third.bin, "w+");

if (!bin)

{

puts("error");

return 1;

}

sortfunc(first, second, third);

return 0;

}

//bilet 12 Даны текстовый и бинарный файлы. Диапазон чисел в этих файлах от 1 до 999

void delfunc(FILE \*txt, FILE \*bin) {

int binary\_number, \* future\_numbers, num = 0;

future\_numbers = (int\*)malloc(2 \* sizeof(int));

if (!future\_numbers) {

exit(error);

}

while (!feof(txt)) {

fscanf\_s(txt, "%d", &future\_numbers[num]);

if (feof(txt))

break;

num++;

future\_numbers = (int\*)realloc(future\_numbers, (num+1) \* sizeof(int));

if (!future\_numbers) {

exit(error);

}

}

num--;

while (!feof(bin)) {

fread(&binary\_number, 1, sizeof(int), bin);

if (feof(bin))

break;

for (int i = 0; i < num+1; i++) {

if (future\_numbers[i] == binary\_number)

future\_numbers[i] = -1;

}

}

fclose(txt);

fopen\_s(&txt, "text.txt", "w+");

rewind(txt);

for (int i = 0; i < num+1; i++) {

if (future\_numbers[i] == -1)

continue;

fprintf(txt, "%3d ", future\_numbers[i]);

}

free(future\_numbers);

return;

}

int main() {

FILE\* bin, \* txt;

fopen\_s(&bin\txt, my\_bin.bin, "w+"); //открываем и меняем название

if (!bin\txt)

{

puts("error");

return 1;

}

delfunc(txt, bin);

return 0;

}

//bilet 13 Из бинарного файла удалить все числа, встречающиеся

void delete(FILE\* txt, FILE\* bin) {

int\* massiv,temp,counter = 0;

massiv = (int\*)malloc(sizeof(int));

if (!massiv) {

exit(error);

}

rewind(bin); rewind(txt);

while (1) {

fread(&temp, sizeof(int), 1, bin);

if (feof(bin))

break;

massiv[counter] = temp;

counter++;

massiv = (int\*)realloc(massiv,(counter + 1) \* sizeof(int));

}

while (1) {

fscanf\_s(txt, "%d", &temp);

if (feof(txt))

break;

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

if (temp == massiv[i]) {

for (int j = i; j < counter - 1; j++)

massiv[j] = massiv[j + 1];

counter--;

massiv = (int\*)realloc(massiv, (counter + 1) \* sizeof(int));

}

}

fclose(bin); fopen\_s(&bin, "binary.bin", "w+");

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

fwrite((massiv + i), sizeof(int), 1, bin);

free(massiv);

}

int main() {

FILE\* bin, \* txt;

fopen\_s(&bin\txt, bin\txt.bin, "w+");

if (!bin || !txt){

puts("error");

return 1;

}

delete(txt, bin);

fclose(bin);

fclose(txt);

return 0;

}

//bilet 14 Есть бинарный файл, упорядоченный по возрастанию

void ent(FILE\* bin) {

printf("entering number");

int enter, checker;

checker = scanf\_s("%d", &enter);

while (!checker)

checker = scanf\_s("%d", &enter);

fpos\_t end;

rewind(bin);

fseek(bin, 0, 2);

fgetpos(bin, &end);

end -= sizeof(int);

if (end <= 0) {

fwrite(&enter, sizeof(int), 1, bin);

return;

}

rewind(bin);

fpos\_t probeg; int temp;

fgetpos(bin, &probeg);

while (probeg <= end) {

fread(&temp, sizeof(int), 1, bin);

if (temp >= enter) {

while (probeg <= end+sizeof(int)) {

int next;

fsetpos(bin, &probeg);

fread(&next, sizeof(int), 1, bin);

fsetpos(bin, &probeg);

fwrite(&enter, sizeof(int), 1, bin);

enter = next;

probeg += sizeof(int);

}

return;

}

else {

probeg += sizeof(int);

continue;

}

}

fwrite(&enter, sizeof(int), 1, bin);

}

int main() {

FILE\* bin;

fopen\_s(&bin, "binary.bin", "r+")

if (!bin\txt) {

puts("error");

return 1;

}

ent(bin);

fclose(bin);

return 0;

}

//bilet 15 Дан текстовый файл, содержащий двузначные числа

void swap(FILE\* txt) {

fpos\_t left = 0, right;

fseek(txt, 0, 2);

fgetpos(txt, &right);

right -= 3;

if (left == right && left == 0) {

printf("file is empty!\n");

return;

}

rewind(txt);

int left\_num, right\_num;

while (left != right) {

fsetpos(txt, &left);

fscanf\_s(txt, "%d", &left\_num);

fsetpos(txt, &right);

fscanf\_s(txt, "%d", &right\_num);

fsetpos(txt, &right);

fprintf(txt,"%2d", left\_num);

fsetpos(txt, &left);

fprintf(txt, "%2d", right\_num);

left += 3;

right -= 3;

if (left > right)

break;

}

return;

}

int main() {

FILE\* txt;

fopen\_s(&txt, "binary.bin", "w+")

if (!txt) {

puts("error");

return 1;

}

swap(txt);

fclose(txt);

return 0;

}

//bilet 16

//В 1 бит мы можем записать только 1 или 0, ост при делении на 2

//в два бита мы можем записать 0 1 2 3 - ост при делении на 4

struct pole {

unsigned int four : 2;

unsigned int two : 1;

int num;

};

int main() {

printf("%d", sizeof(pole));

pole \*a;

a = (pole\*)malloc(sizeof(pole));

scanf\_s("%d", &a->num);

//присваиваются только первые биты

a->four = a->num;

a->two = a->num;

printf("four: %d\ntwo: %d\n", a->four, a->two);

}

//Билет 17 Рекурсивно создать двунаправленное кольцо

struct ring {

int n;

ring\* left;

ring\* right;

};

void create(ring\*\* head, int n) {

if (n == 0) {

return;

}

ring\* s;

s = (ring\*)calloc(1, sizeof(ring));

printf("Введите число\n");

while (!scanf("%d", &s->n)) {

rewind(stdin);

printf("Error\n");

}

s->left = s->right = nullptr;

if (!(\*head)) {

s->left = s->right = s;

(\*head) = s;

}

else {// если кольцо уже есть

s->left = (\*head)->left;

(\*head)->left->right = s;

s->right = (\*head);

(\*head)->left = s;

}

create(head, n - 1);

}

int main() {

ring\* head;

head = nullptr;

create(&head, 5);

return 0;

}

//Билет 18 Записать в битовом представлении буквы алфавита

struct pole {

unsigned int i1 : 1, i2 : 1, i3 : 1, i4 : 1, i5 : 1, i6 : 1, i7 : 1, i8 : 1;

// каждой переменной присваиваем значение одного бита

};

int main() {

pole\* c;

char symbal = 'a'; // начнём с а и в цикле пройдём по всему алфавиту

for (int i = 0; i < 26; i++) { // 26 букв в английском языке

c = (pole\*)&symbal; //это желательно запомнить

symbal = symbal + 1; // просто увеличиваем аскикод чтобы перейти к следующей букве

}

return 0;

}

//Билет 19 Удаление элемента из однонаправленной очереди.

struct queu {м

int n;

queu\* next;

};

void delette(queu\*\* head, queu\*\* tail, int n) {

queu\* buf, \* ptr;

int i = 1;

buf = (\*head);

ptr = buf->next;

while (1) {

if (n == 0) {

(\*head) = (\*head)->next;

free(buf);

break;

}

else if (!ptr->next && n==i) {

(\*tail) = buf;

buf->next = nullptr;

free(ptr);

break;

}

else if (i == n) {

buf->next = ptr->next;

free(ptr);

break;

}

i++;

ptr = ptr->next; buf = buf->next;

}

}

int main() {

queu\* head, \* tail;

head = tail = nullptr;

int n;

printf("Номер для удаления\n");

while (!(scanf("%d", &n) && n>0)) {

printf("error\n");

rewind(stdin);

}

delette(&head, &tail, n-1);

return 0;

}

//Билет 20 Dля k-ого элемента: если левое поддерево, то 2k правое, то 2k + 1

void add(int\*\* derevo, int& numeretic, bool\*\* set) {

int enter\_number;

while (!scanf("%d", &enter\_number) {

printf("error\n");

rewind(stdin);

}

int k = 1;

while (1) {

if (k > numeretic) {

numeretic = k;

(\*derevo) = (int\*)realloc((\*derevo), (numeretic + 1) \* sizeof(int));

(\*set) = (bool\*)realloc((\*set), (numeretic + 1) \* sizeof(bool));

continue;

}

if ((\*derevo)[k] == enter\_number) {

printf("number already exist!\n");

system("pause");

return;

}

if ((\*set)[k]) {

(\*derevo)[k] = enter\_number;

(\*set)[k] = false;

break;

}

else if (enter\_number < (\*derevo)[k]) {

k \*= 2;

continue;

}

else {

k = k \* 2 + 1;

continue;

}

}

}

int main() {

int\* derevo, numeretic = 1;

bool\* set;

derevo = (int\*)malloc((numeretic + 2) \* sizeof(int));

if (!derevo) {

exit(error);

}

set = (bool\*)malloc((numeretic + 2) \* sizeof(bool));

if (!set) {

exit(error);

}

add(&derevo, numeretic, &set);

return 0;

}

//Билет 21 Даны массивы структур из двух элементов Найти студента с наибольшим баллом.

struct inf {

char name[30];

int group;

};

struct mark {

int mark;

};

struct student {

union {

inf a;

mark b;

}un;

};

int main() {

student\*\* mas;

int n;

char name[30];

mas = (student\*\*)calloc(n, sizeof(student\*));

if (!mas) {

exit(error);

}

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

mas[i] = (student\*)calloc(2, sizeof(student));

}

int max = 0, maxi;

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

if (mas[i][1].un.b.mark > max) {

max = mas[i][1].un.b.mark;

maxi = i;

}

}

printf("\nСтудент с наивысшей отметкой\n");

printf("Имя %s\n", mas[maxi][0].un.a.name);

printf("Группа %d\n", mas[maxi][0].un.a.group);

printf("Отметка %d\n", mas[maxi][1].un.b.mark);

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

free(mas[i]);

}

return 0;

}

//Билет 22 Организовать ввод числа с клавиатуры и удаление из бинарного файла.

void del(FILE\* f, int n) {

fpos\_t start, end;

int i,n1 = 0;

int i1;

int size = ftell(f);

rewind(f);

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

while (1) {

if (n1 == n) {

fseek(f, -4, 1);

fgetpos(f, &start);

fread(&i1, sizeof(int), 1, f);

fgetpos(f, &end);

while (1) {

fsetpos(f, &end);

fread(&i1, sizeof(int), 1, f);

if (feof(f)) {

break;

}

fsetpos(f, &start);

fwrite(&i1, sizeof(int), 1, f);

start++;

end++;

}

chsize(fileno(f), size - 4);

break;

}

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

if (feof(f)) {

break;

}

n1++;

}

}

int main() {

FILE\* f;

if (!(f = fopen("file", "w+b"))) {

printf("Error\n");

return 0;

}

int n;

int i1;

printf("сколько чисел добавить?\n");

while (!(scanf("%d", &n) && n<0)) {

printf("error\n");

rewind(stdin);

}

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

{

printf("Введите число\n");

while ((!scanf("%d", &i1) && n<0)) {

rewind(stdin);

}

fwrite(&i1, sizeof(int), 1, f);

}

printf("\nВведите номер числа которое хотите удалить\n");

while (!(scanf("%d", &n) && n<0)) {

rewind(stdin);

}

del(f, n-1);

fclose(f);

return 0;

}

//Билет 23 добавления русского символа, не нарушая упорядоченности.

int main() {

system("chcp 1251>null");

char\* string;

int flag =0;

string = (char\*)calloc(100, sizeof(char));

if (!string) {

exit(error);

}

gets\_s(string, 100);

FILE\* f;

if (!(f = fopen("file.txt", "w+"))) {

printf("Error\n");

return 0;

}

rewind(f);

char c,c1, c2;

printf("\enter rus symb\n");

char symbol;

while (!(scanf("%c", &symbol) && n<0)) {

printf("error\n");

rewind(stdin);

}

rewind(stdin);

rewind(f);

while (1) {

while (1) {

fscanf(f, "%c", &c);

if (feof(f)) {

break;

}

if (symbol < c && c != ' ') {

fseek(f, -1, 1);

flag = 1;

break;

}

}

if (flag == 0){

fprintf(f, "%c", symbol);

break;

}

else {

c1 = c;

fprintf(f, "%c", symbol);

while (1) {

fscanf(f, "%c", &c2);

if (feof(f)) {

fprintf(f, "%c", c1);

break;

}

fseek(f, -1, 1);

fprintf(f, "%c", c1);

c1 = c2;

}

break;

}

}

rewind(f);

fclose(f);

return 0;

}