Министерство Образования и Науки Российской Федерации  
Новосибирский Государственный Технический Университет  
Кафедра Прикладной Математики

**Лабораторная работа № 5**

**по дисциплине «Структуры данных и алгоритмы»**

Факультет: ФПМИ

Группа: ПМ-53

Студент: Тябин Егор Алексеевич

Преподаватель: Еланцева Ирина Леонидовна

### Новосибирск 2016

# Условие задачи

В файле «Авто» дана информация о автомобилях: <номер, а/м>, <марка, а/м>, <ФИО владельца>. Создать таблицу «Автовладельцы», с элементами того же вида. Таблица должна быть упорядочена по возрастанию <номер а/м>.  
Таблицу «Автовладельцы» (созданную в л/р 4) отсортировать методом пирамидальной сортировки. Используя таблицу «Аварии» создать таблицу предполагаемых владельцев угнанной машины, о которой известна только марка. Элементы таблицы: <номер>, <ФИО владельца>.

# 2. Анализ задачи

*Дано:*   
В файле «Авто»  
Последовательность строк, где в каждой строке находятся следующие последовательности символов:  
<номер а/м> состоящий из цифр и букв, количество символов меньше 14.  
<марка а/м> состоящая из букв, количество символов меньше 30.  
<Фамилия> состоящая из букв, количество символов меньше 20.  
<Имя> состоящая из букв, количество символов меньше 20.  
<Отчество> состоящая из букв, количество символов меньше 20.  
Последовательности символов разделяются пробелом, в конце строки пробел не ставится.

В файле «Аварии»  
Слово:  
<марка а/м> состоящая из букв, количество символов меньше 30.  
*Результат:*   
Таблица – совокупность массива структурного типа FIO и массива символов(слово) <номер>. Тип поля FIO в свою очередь, является совокупностью трех массивов символьного типа, где максимальное количество элементов массивов не превышает 20.

## Метод решения

Для работы с нашей задачей выделим несколько основных подзадач.

## Выделение основных подзадач

1. Сортировка массива методом пирамидальной сортировки: создание сортирующего дерева
2. Сортировка массива методом пирамидальной сортировки
3. Поиск и вывод автовладельцев с автомобилем заданной марки

1) Сортировка массива методом пирамидальной сортировки: создание сортирующего дерева.  
Дан массив, представляя его в виде дерева, делаем сортирующее дерево. Для этого делим на 2 части массив, этим действием подразумевая разделение на поддеревья и листья. Далее будем расширять часть массива, обладающую столь полезным свойством, добавляя по одному элементу за шаг. Следующий элемент на каждом шаге добавления - тот, который стоит перед уже готовой частью.

Чтобы при добавлении элемента сохранялась пирамидальность, будем использовать следующую процедуру расширения пирамиды a[i+1] … a[n] на элемент a[i] влево:

1. Смотрим на сыновей слева и справа - в массиве это a[2i+1] и a[2i+2] и выбираем наибольшего из них.
2. Если этот элемент больше a[i] - меняем его с a[i] местами и идем к шагу 1, имея в виду новое положение a[i] в массиве. Иначе конец процедуры.

Новый элемент "просеивается" сквозь пирамиду. Выполняется "свойство пирамиды": каждый элемент меньше, либо равен родителю.

2) Сортировка массива методом пирамидальной сортировки.   
Итак, задача построения пирамиды из массива успешно решена. Как видно из свойств пирамиды, в корне всегда находится максимальный элемент. Отсюда вытекает алгоритм фазы 2:

1. Берем верхний элемент пирамиды a [0] ... a[n] (первый в массиве) и меняем с последним местами. Теперь "забываем" об этом элементе и далее рассматриваем массив a [0] ... a[n-1]. Для превращения его в пирамиду достаточно просеять лишь новый первый элемент.
2. Повторяем шаг 1, пока обрабатываемая часть массива не уменьшится до одного элемента.

Очевидно, в конец массива каждый раз попадает максимальный элемент из текущей пирамиды, поэтому в правой части постепенно возникает упорядоченная последовательность.

3) Поиск и вывод автовладельцев с автомобилем заданной марки.  
Считывается марка автомобиля, после чего происходит поиск автовладельцев с такой же маркой перебором всех автовладельцев.

# Структуры данных, используемые для представления исходных данных и результатов задачи

## Внешнее представление входных данных

Последовательность слов.

## Внешнее представление выходных данных

Последовательность слов, выводимых на экране в ходе работы программы.

# Внутреннее представление входных данных

Struct table{ ahuman human[19]; int n; } – структура table, состоящая из 2-х полей: структуры ahuman и целочисленного значения n. Другими словами это линейный массив с указателем на конец.

Char mark[30] – массив символов. В которую заносится марка автомобиля.

## Внутреннее представление данных

struct ahuman{ char number[14], mark[30]; FIO name; } - структура ahuman, состоящая из 3-х полей: массивов number и mark символьного типа и name типа FIO.   
struct FIO{ char F[20], I[20], O[20]; } – структура FIO, состоящая из 3-х полей: массивов F, I и O символьного типа.  
struct tablem{ FIO thief[19]; int n; } – структура tablem(таблица), состоящая из 2-х полей: структуры FIO и целочисленного значения n. Другими словами это линейный массив с указателем на конец.  
struct bhuman {char number[14], FIO name;} – структура bhuman, состоящая из 2-х полей: массива number символьного типа и name типа FIO.

table avtovladelcy – таблица «Автовладельцы».

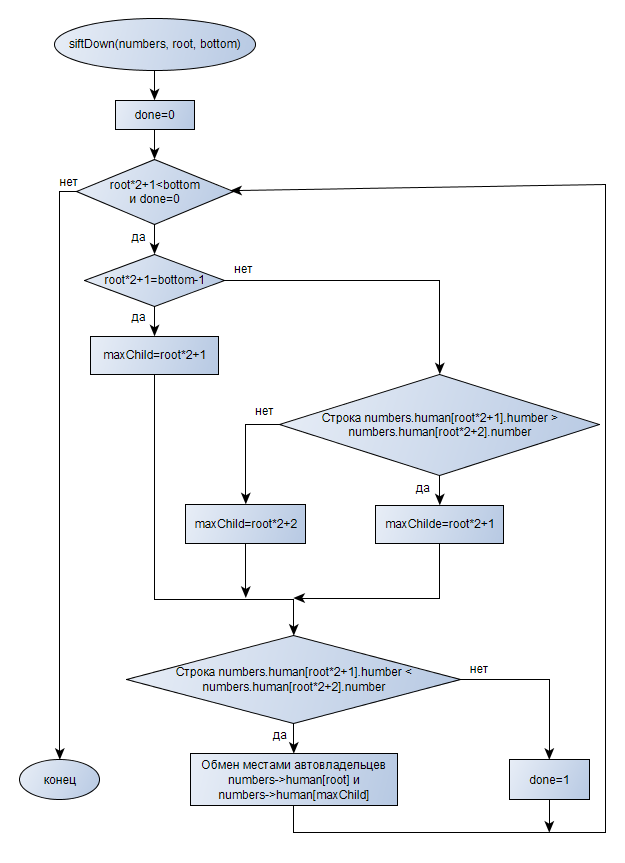
ahuman man – элемент типа ahuman в который записывается значения из файла «Авто».

tablem suspects – таблица автовладельцев, имеющих автомобили с заданной маркой.

FILE \*fp, \*fpp, \*fpm – указатели на поток файлов «Авто», «Аварии» и «Подозреваемые» соответственно.

int i, j – целочисленные элементы, используемые в программе для перечисления элементов.  
int done – целочисленный элемент, используется в подпрограмме siftDown для выхода из цикла замены.  
int maxChild – целочисленное значение обозначающее номер максимального элемента поддерева.  
ahuman temp – элемент типа ahuman использующийся для записи в него промежуточного значения в процессе замены двух элементов.

# C:\Users\Egor\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2016-05-26_16-55-48.pngC:\Users\Egor\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2016-05-26_17-07-15.pngУкрупненный алгоритм решения задачи



# Структура программы

## Взаимосвязь функций

Void main()

void heapSort(table \*numbers);

## Составные части программы

Наименование функции:   
heapSort  
Прототип функции:  
void heapSort(table \*numbers);  
Данная подпрограмма контролирует 2 фазы пирамидальной сортировки. Сначала контролируя создание сортирующего дерева, а потом саму сортировку.

Наименование функции:   
siftDown  
Прототип функции:  
void siftDown(table \*numbers, int root, int bottom);  
Данная подпрограмма контролирует отвечает за сравнение определенных элементов дерева в зависимости от фазы сортировки и перестановки нужных элементов.

Наименование функции:  
review  
Прототип функции:  
void review(FILE \*fpr, FILE \*fpn, tablem \*T, table L);  
Данная подпрограмма создает таблицу «Suspects» путём отбора автовладельцев с заданной маркой и попутно выписывает в файл «Подозреваемые» номер и ФИО владельца машины с данной маркой.

# Текст программы

**/\* Файл tables.h \*/**  
#ifndef TABLES\_H

#define TABLES\_H

struct FIO{ char F[20], I[20], O[20]; };

struct ahuman{ char number[14], mark[30]; FIO name; };

struct bhuman{ char number[14]; FIO name; };

struct table{ ahuman human[19]; int n; };

struct tablem{ bhuman thief[19]; int n; };

void review(FILE \*fpr, FILE \*fpn, tablem \*T, table L);

void siftDown(table \*numbers, int root, int bottom);

void heapSort(table \*numbers);

#endif

**/\* Файл tables.cpp \*/**  
#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include "tables.h"

#include <string.h>

void review(FILE \*fpr, FILE \*fpn, tablem \*T, table L){

char mark[30];

int i, j = 0;

if (fscanf(fpr, "%s", &mark) != EOF)

{

for (i = 0; i < L.n; i++){

if (strcmp(mark, L.human[i].mark) == 0) {

T->thief[j].name = L.human[i].name;

strcpy(T->thief[j].number, L.human[i].number);

fprintf(fpn, "%s %s %s %s\n", T->thief[j].number, T->thief[j].name.F, T->thief[j].name.I, T->thief[j].name.O);

j++;

T->n = j;

}

}

}

}

void siftDown(table \*numbers, int root, int bottom) {

int done, maxChild;

ahuman temp;

done = 0;

while ((root \* 2+1 < bottom) && (!done)) {

if (root \* 2+1 == bottom-1)

maxChild = root \* 2+1;

else if (strcmp(numbers->human[root \* 2+1].number, numbers->human[root \* 2 + 2].number)>0)

maxChild = root \* 2+1;

else

maxChild = root \* 2 + 2;

if (strcmp(numbers->human[root].number, numbers->human[maxChild].number)<0) {

temp = numbers->human[root];

numbers->human[root] = numbers->human[maxChild];

numbers->human[maxChild] = temp;

root = maxChild;

}

else

done = 1;

}

}

void heapSort(table \*numbers) {

int i;

ahuman temp;

for (i = (numbers->n / 2) - 1; i >= 0; i--)

siftDown(numbers, i, numbers->n);

for (i = numbers->n - 1; i >= 1; i--) {

temp = numbers->human[0];

numbers->human[0] = numbers->human[i];

numbers->human[i] = temp;

siftDown(numbers, 0, i);

}

}

**/\* Файл main.cpp \*/**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

#include "tables.h"

void main(){

table avtovladelcy;

ahuman man;

tablem suspects;

setlocale(LC\_CTYPE, "Russian");

FILE \*fp, \*fpp, \*fpm;

fp = fopen("Авто.txt", "r");

fpp = fopen("Аварии.txt", "r");

fpm = fopen("Подозреваемые.txt", "w");

int i = 0;

avtovladelcy.n = 0;

while (fscanf(fp, "%s", &man.number) != EOF && avtovladelcy.n<19){

fscanf(fp, "%s%s%s%s", &man.mark, &man.name.F, &man.name.I, &man.name.O);

avtovladelcy.human[i] = man;

avtovladelcy.n++;

i++;

}

heapSort(&avtovladelcy);

review(fpp, fpm, &suspects, avtovladelcy);

fclose(fp);

fclose(fpp);

fclose(fpm);

}

# 7.Тесты

**Тест 1.**  
Назначение: проверка на все типы упорядоченности и марку которой нет в «Авто».  
Входные данные:   
/\* Авто \*/  
кв215а54 BMW Тябин Егор Алексеевич

уг987ия154 TOYOTA Иванов Иван Иванович

таб23ли76 ROLS-ROIS Подозрительной Внешности Человек

ав732ку154 VOLGA Евангелион Супрем Финихтович

бг987ре32 RHINO Филипс Тревор Безбашенный

ята876н59 MERCEDES Куваль Огнетушитель Фридирхович

/\* Аварии \*/  
RANGE-ROVER

Результат:

**Тест 2.**  
Назначение: Если файл «Аварии» пустой.  
Входные данные:  
/\* Авто \*/  
кв215а54 BMW Тябин Егор Алексеевич

уг987ия154 TOYOTA Иванов Иван Иванович

таб23ли76 ROLS-ROIS Подозрительной Внешности Человек

ав732ку154 VOLGA Евангелион Супрем Финихтович

бг987ре32 RHINO Филипс Тревор Безбашенный

ята876н59 MERCEDES Куваль Огнетушитель Фридирхович

Результат:

**Тест 3.**  
Назначение: Проверка на работоспособность  
Входные данные:  
/\* Авто \*/  
кв215а54 BMW Тябин Егор Алексеевич

уг987ия154 TOYOTA Иванов Иван Иванович

цек723к86 VOLGA Аргыз Тутнхейн Пятый

таб23ли76 ROLS-ROIS Подозрительной Внешности Человек

ав732ку154 VOLGA Евангелион Супрем Финихтович

бг987ре32 RHINO Филипс Тревор Безбашенный

ята876н59 MERCEDES Куваль Огнетушитель Фридирхович

/\* Аварии \*/

VOLGA

Результат:

ав732ку154 Евангелион Супрем Финихтович

цек723к86 Аргыз Тутнхейн Пятый

# 8. Результат работы программы

-Программа выдала верное решение на всех тестах и, следовательно, правильно работает.