Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Кубанский государственный университет»

Факультет компьютерных технологий и прикладной математики

**Кафедра информационных технологий**

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

**(практика по получению первичных профессиональных умений и навыков)**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.А.Казимир

студентка ФКТиПМ, 1 курса, 11 гр.,

направление 01.03.02 «Прикладная математика и информатика»

Руководитель учебной практики

(практика по получению

первичных профессиональных

умений и навыков) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.П.Лукащик

к.ф.-м.н., доцент кафедры

информационных технологий

Краснодар 2019

* 1. **Условие задачи:**

Тестирование проводится по трем предметам, по каждому предмету абитуриент может набрать от 0 до 100 баллов. При этом к сдаче экзаменов в первом потоке допускаются абитуриенты, набравшие по результатам тестирования не менее 30 баллов по каждому из трёх предметов, причём сумма баллов должна быть не менее 140. Напишите программу, которая будет сохранять в выводном файле фамилии и имена абитуриентов, допущенных к сдаче экзаменов в первом потоке. При этом фамилии должны выводиться в алфавитном порядке.

* 1. **Описание алгоритма решения и структур хранения данных:**

В программе используются следующие структуры:

1. Структура student с полями типа string: surname(фамилия студента) и name(имя студента),типа int: ex1,ex2,ex3 (баллы по трём предметам). Будет использована в качестве буфера для обработки данных.
2. Структура tree с полем root типа student (корень дерева), полями L и R типа tree\*(указатели на левое и правое поддерево соответственно). Будет использована в качестве двоичного дерева поиска для записи в него результата, сортировки и вывода.

Алгоритм.

Процедура makeinput создаёт текстовый файл input.txt, в который записываются входные данные с консоли. Для этого из консоли считывается количество элементов, затем поочерёдно считываются сами элементы.

Процедура использует следующие переменные:

1. in (ofstream) – создание и открытие файла input.txt для записи.
2. n (int) – количество вводимых элементов.
3. buf (student\*) – буфер для элементов.

Процедура makebin создаёт бинарный файл binary.bin и переписывает в него данные из input.txt.

Процедура использует следующие переменные:

1. in (ifstream) – открытие файла input.txt для чтения.
2. bin (ofstream) – создание и открытие файла binary.txt для записи.
3. buf (student\*) – буфер для элементов.

Процедура add добавляет элемент buf (student\*) в поддерево с корнем t (tree\*).Распределение элементов по дереву проводится в соответствии со значениями surname и name по алфавиту.

Процедура использует следующие переменные:

1. tmp (tree\*)-инициализируется для добавления элемента в дерево.
2. buf (student\*) – добавляемый элемент.
3. t (tree\*) – указатель на корень дерева.

Функция formtree возвращает двоичное дерево поиска t (tree\*) , содержащее элементы, удовлетворяющие условию задачи, отсортированные по значениям surname и name по алфавиту. Для этого открывается файл binary.txt, и оттуда считываются элементы пока не кончится файл. Каждый элемент записывается в буфер buf, в затем проверяется на условия задачи. В случае, если элемент удовлетворяет условиям задачи он добавляется в дерево t с помощью процедуры add.

Функция использует следующие переменные:

1. t (tree\*) – указатель на корень создаваемого дерева.
2. bin (ifstream) – открытие файла binary.txt для чтения.
3. buf (student\*) – буфер для элементов.

Процедура makeoutput выводит фамилии и имена, удовлетворяющие условию задачи, в алфавитном порядке из двоичного дерева поиска t на экран консоли и в файл output.txt.

Процедура использует следующие переменные:

1. t (tree\*) – указатель на корень дерева.
2. out (ofstream) – для записи в файл output.txt.

В функции main выполняются следующие шаги алгоритма:

1. Вызывается функция makeinput.
2. Вызывается функция Makebin.
3. Создаётся переменная t (\*tree) и инициализируется функцией formtree.
4. Создаётся и открывается для записи текстовый файл output.txt через переменную out.
5. Вызывается функция makeoutput от переменных t и out.
	1. **Описаниевходныхивыходныхданных*:***

Ввод данных осуществляется из текстового файла(input.txt). Каждая запись файла имеет следующий формат:

**<Фамилия> <Имя> <Баллы>**

**Пример: Иванов Иван 12 34 56**

Здесь <Фамилия> – строка, состоящая не более чем из 20 символов; <Имя> – строка, состоящая не более чем из 15 символов; <Баллы> – три целых числа, соответствующих баллам, полученным на тестировании по каждому из трех предметов.

В основной программе происходит считывание каждой строки текстового файла, занесение значений в соответствующие поля массива структур, сортировка и вывод в выводной файл(output.txt) фамилий и имен абитуриентов, допущенных к сдаче экзаменов в первом потоке. При этом фамилии выводится в алфавитном порядке.

* 1. **Текст программы:**

#include <iostream>

#include <string>

 #include <fstream>

using namespace std;

struct student {

 string surname, name;

 int ex1, ex2, ex3;

};

struct tree {

 student root;

 tree\* L, \*R;

};

void makeinput() {

 student\* buf = new student;

 int n;

 ofstream in("input.txt", ios::binary);

 cin >> n;

 while (n) {

 cin >> buf->surname >> buf->name >> buf->ex1 >> buf->ex2 >> buf->ex3;

in << buf->surname << " " << buf->name << " " << buf->ex1 << " " << buf->ex2 << " " << buf->ex3;

 if (n > 1) in << endl;

 n--;

 }

 in.close();

}

void makebin() {

 ifstream in("input.txt");

 ofstream bin("binary.bin", ios::binary);

 student\* buf = new student;

 while (!in.eof()) {

 in >> buf->surname >> buf->name >> buf->ex1 >> buf->ex2 >> buf->ex3;

 bin.write((char\*)buf, sizeof(student));

 }

 in.close();

 bin.close();

}

void add(tree\* & t, student \* buf) {

 if (t == 0)

 {

 t = new tree;

 t->root = \*buf;

 t->L = 0;

 t->R = 0;

 }

 else {

 if ((buf->surname < t->root.surname) || ((buf->surname == t->root.surname) && (buf->name < t->root.name)))

 add(t->L, buf);

 else add(t->R, buf);

 }

}

tree\* formtree() {

 tree\*t = 0;

 ifstream bin("binary.bin", ios::binary);

 student\* buf = new student;

 while (!bin.eof()) {

 bin.read((char\*)buf, sizeof(student));

 if ((buf->ex1 >= 30) && (buf->ex2 >= 30) && (buf->ex3 >= 30) && ((buf->ex1 + buf->ex2 + buf->ex3) >= 140))

 add(t, buf);

 }

 bin.close();

 return t;

}

void makeoutput(tree\*t, ofstream &out) {

 if (t) {

 makeoutput(t->L, out);

 out << t->root.surname << " " << t->root.name << endl;

 cout << t->root.surname << " " << t->root.name << endl;

 makeoutput(t->R, out);

 }

}

int main()

{

 cout << "input" << endl;

 makeinput();//создаём текстовый файл input.txt и вводим в него данные с консоли

 makebin();//создаём бинарный файл binary.bin и копируем в него данные из input.txt

 tree\*t = formtree();//обрабатываем данные из input txt, помещаем результат в двоичное дерево поиска t

 cout << endl << "output" << endl;

 ofstream out("output.txt");//создаём текстовый файл output.txt

 makeoutput(t, out);//записываем результат из двоичного деерва поиска t в файл output.txt, а также выводим данные в консоль

 out.close();

 system("pause");

 return 0;

}

* 1. **Тестовый запуск программы и вывод результатов работы:**

Вводим количество элементов: 6

 Вводим с клавиатуры элементы:

A A 30 30 30

A B 29 100 100

C A 100 100 100

A C 50 50 50

B D 70 70 70

B C 60 60 60

 Содержимое input.txt:

A A 30 30 30

A B 29 100 100

C A 100 100 100

A C 50 50 50

B D 70 70 70

B C 60 60 60

Выведенный на экран результат:

A C

B C

B D

C A

Содержимое output.txt:

A C

B C

B D

C A

Программа работает корректно.