Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе 8**

Дисциплина: Основы алгоритмизации и программирования

Выполнил:

Студентка группы ИВТ-22-2б

Мифтахов Марат Ринатович

Проверил:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2023**

**Постановка задачи**

Структура "Студент":

- фамилия, имя, отчество;

- номер телефона;

- группа;

- оценки по 3 основным предметам.

Удалить все элементы, у которых среднее арифметическое оценок меньше заданного, добавить элемент после элемента с заданной фамилией.

**Анализ задачи**

1. Вводится количество студентов, создается динамический массив, в котором будут храниться данные о студентах.
2. Записываем студентов через цикл, в котором вызывается функция show(), для записи данных о студентах.

void show()

{

cout << endl;

cout << "Surname: " << this->surname << endl;

cout << "Name: " << this->name << endl;

cout << "Patr: " << this->patr << endl;

cout << "Phone: " << this->phone << endl;

cout << "Group: " << this->group << endl;

cout << "Marks: " << this->mark[0] << ", " << this->mark[1] << ", " << this->mark[2] << endl;

}

1. В этом же цикле вызывается функция average(), которая считает среднее арифметическое оценок каждого студента.

double average()

{

double r = 0;

for (int i = 0; i < 3; i++)

{

r += this->mark[i];

}

return r / 3;

};

1. Далее на ввод от пользователя получаем среднее арифметическое и вызывается функция DeleteLowerAverage(students, N, average)

void DeleteLowerAverage(Student\*& mas, int& size, const double &value)

{

int tmp\_size = 0;

Student\* tmp = new Student[tmp\_size];

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

{

if (mas[i].average() >= value)

{

PushBack(tmp,tmp\_size,mas[i]);

}

}

delete[] mas;

mas = tmp;

size = tmp\_size;

};

1. Если среднее арифметическое оценок студента больше или равно среднему арифметическому, введенному пользователем, то вызывается функция Pushback (tmp, tmp, size, mas[i])

void PushBack(T\*& mas, int& size, const T& value)

{

T\* tmp = new T[size + 1];

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

{

tmp[i] = mas[i];

}

tmp[size] = value;

delete[] mas;

mas = tmp;

size++;

};

1. После удаления студента, в консоль выводятся все остальные через цикл, в котором вызвана функция show().
2. Затем на ввод от пользователя получаем фамилию студента, после которого нужно добавить данные о новом студенте, если фамилии студента нет, выводится сообщение об ошибке, а если есть, то вызывается функция InsertAt()

void InsertAt(T\*& mas, int& N, const int& index, const T& value)

{

T\* result\_mas = new T[N + 1];

int c = 0;

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

{

result\_mas[c++] = mas[i];

}

result\_mas[c++] = value;

for (int i = index; i < N; i++)

{

result\_mas[c++] = mas[i];

}

delete[] mas;

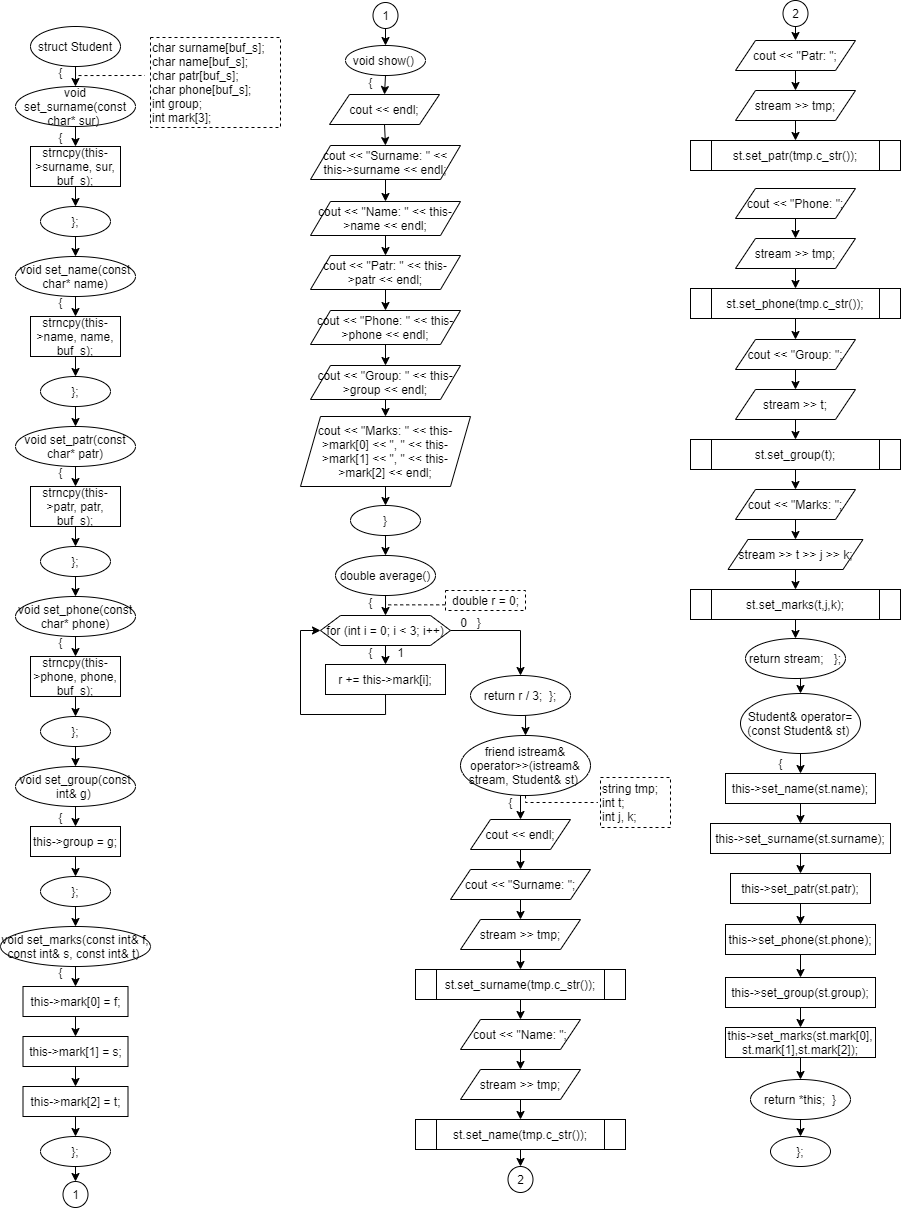
mas = result\_mas;

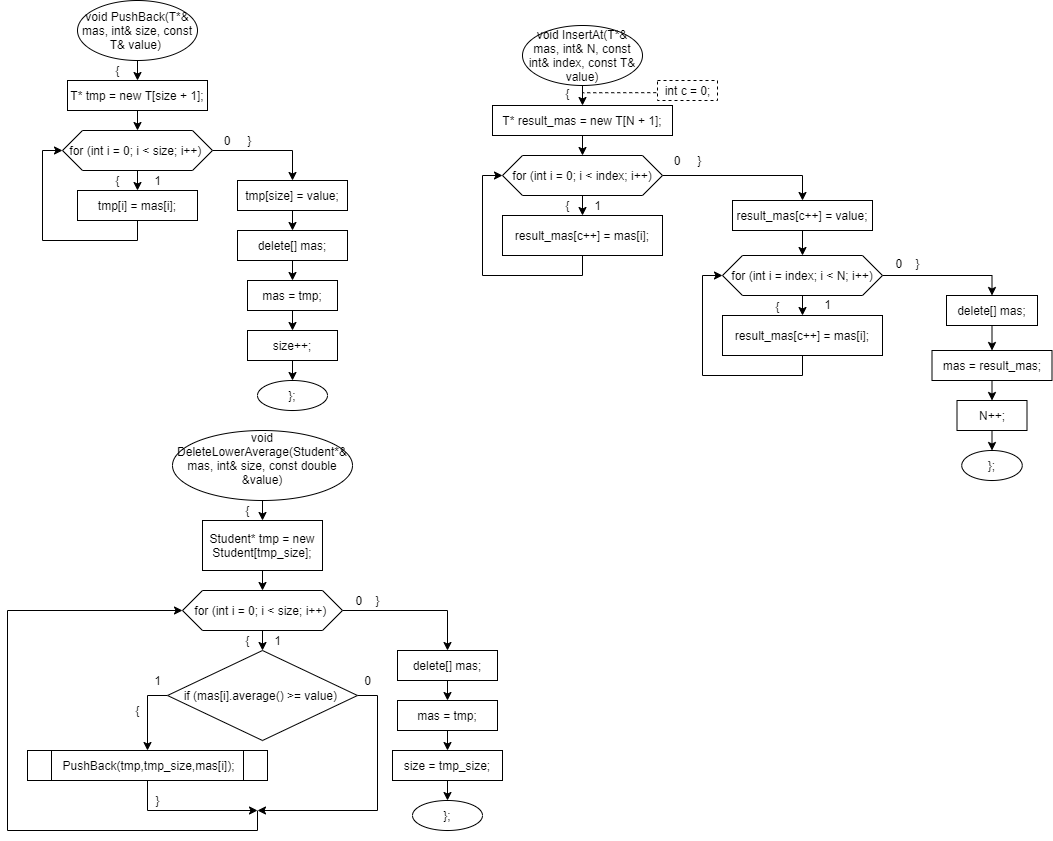
N++;

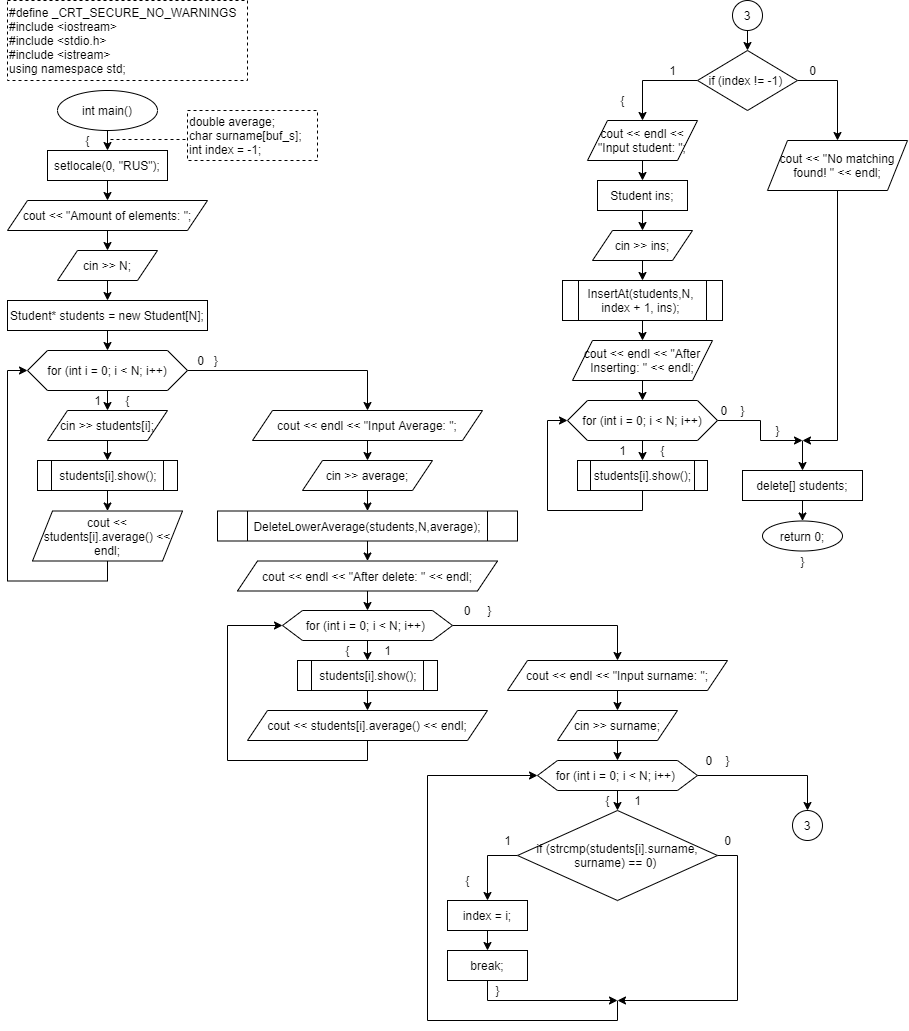
};

1. После добавления студента, выводится полный список студентов в консоль.

**Блок-схема**







**Исходный код**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <istream>

using namespace std;

const int buf\_s = 32;

struct Student

{

char surname[buf\_s];

char name[buf\_s];

char patr[buf\_s];

char phone[buf\_s];

int group;

int mark[3];

void set\_surname(const char\* sur)

{

strncpy(this->surname, sur, buf\_s);

};

void set\_name(const char\* name)

{

strncpy(this->name, name, buf\_s);

};

void set\_patr(const char\* patr)

{

strncpy(this->patr, patr, buf\_s);

};

void set\_phone(const char\* phone)

{

strncpy(this->phone, phone, buf\_s);

};

void set\_group(const int& g)

{

this->group = g;

};

void set\_marks(const int& f, const int& s, const int& t)

{

this->mark[0] = f;

this->mark[1] = s;

this->mark[2] = t;

};

void show()

{

cout << endl;

cout << "Surname: " << this->surname << endl;

cout << "Name: " << this->name << endl;

cout << "Patr: " << this->patr << endl;

cout << "Phone: " << this->phone << endl;

cout << "Group: " << this->group << endl;

cout << "Marks: " << this->mark[0] << ", " << this->mark[1] << ", " << this->mark[2] << endl;

}

double average()

{

double r = 0;

for (int i = 0; i < 3; i++)

{

r += this->mark[i];

}

return r / 3;

};

friend istream& operator>>(istream& stream, Student& st)

{

string tmp;

cout << endl;

cout << "Surname: ";

stream >> tmp;

st.set\_surname(tmp.c\_str());

cout << "Name: ";

stream >> tmp;

st.set\_name(tmp.c\_str());

cout << "Patr: ";

stream >> tmp;

st.set\_patr(tmp.c\_str());

cout << "Phone: ";

stream >> tmp;

st.set\_phone(tmp.c\_str());

int t;

cout << "Group: ";

stream >> t;

st.set\_group(t);

int j, k;

cout << "Marks: ";

stream >> t >> j >> k;

st.set\_marks(t,j,k);

cout << endl;

return stream;

};

Student& operator=(const Student& st)

{

this->set\_name(st.name);

this->set\_surname(st.surname);

this->set\_patr(st.patr);

this->set\_phone(st.phone);

this->set\_group(st.group);

this->set\_marks(st.mark[0], st.mark[1],st.mark[2]);

return \*this;

}

};

template<class T>

void PushBack(T\*& mas, int& size, const T& value)

{

T\* tmp = new T[size + 1];

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

{

tmp[i] = mas[i];

}

tmp[size] = value;

delete[] mas;

mas = tmp;

size++;

};

void DeleteLowerAverage(Student\*& mas, int& size, const double &value)

{

int tmp\_size = 0;

Student\* tmp = new Student[tmp\_size];

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

{

if (mas[i].average() >= value)

{

PushBack(tmp,tmp\_size,mas[i]);

}

}

delete[] mas;

mas = tmp;

size = tmp\_size;

};

template <class T>

void InsertAt(T\*& mas, int& N, const int& index, const T& value)

{

T\* result\_mas = new T[N + 1];

int c = 0;

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

{

result\_mas[c++] = mas[i];

}

result\_mas[c++] = value;

for (int i = index; i < N; i++)

{

result\_mas[c++] = mas[i];

}

delete[] mas;

mas = result\_mas;

N++;

};

int N;

std::string tmp;

int main()

{

cout << "Amount of elements: ";

cin >> N;

cout << endl;

Student\* students = new Student[N];

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

{

cin >> students[i];

students[i].show();

cout << students[i].average() << endl;

}

double average;

cout << endl << "Input Average: ";

cin >> average;

DeleteLowerAverage(students,N,average);

cout << endl << "After delete: " << endl;

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

{

students[i].show();

cout << students[i].average() << endl;

}

char surname[buf\_s];

cout << endl << "Input surname: ";

cin >> surname;

int index = -1;

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

{

if (strcmp(students[i].surname, surname) == 0)

{

index = i;

break;

}

}

if (index != -1)

{

cout << endl << "Input student: ";

Student ins;

cin >> ins;

cout << endl;

InsertAt(students,N, index + 1, ins);

cout << endl << "After Inserting: " << endl;

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

{

students[i].show();

}

}

else

{

cout << "No matching found! " << endl;

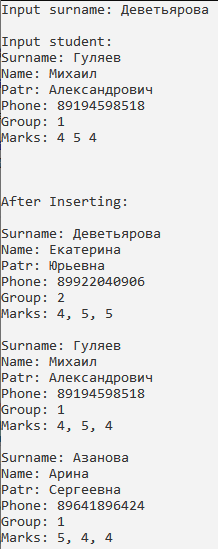
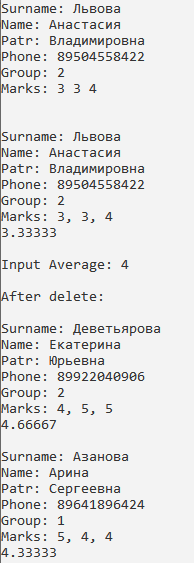
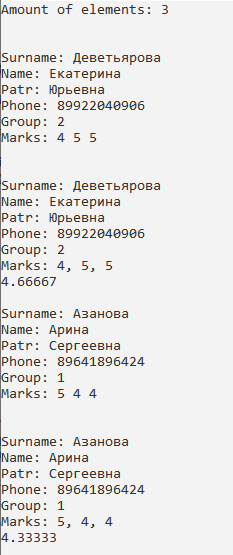
}

delete[] students;

return 0;

}

**Скриншоты программы**

****

**Анализ (объяснение скриншотов)**

Программа запрашивает у пользователя количество студентов, пользователь вводит данные о студентах, программа о каждом из них выводит данные, введенные пользователем, а также выводит среднее арифметическое оценок у каждого студента. Затем программа запрашивает у пользователя среднее арифметическое оценок, выше которого должно быть, далее выводит всех студентов, у которых среднее арифметическое оценок выше или равно заданному пользователем. Далее программа запрашивает у пользователя фамилию студента, после которого нужно добавить студента, если такой студент есть, то программа запрашивает у пользователя данные о новом студенте и выводит всех студентов, включая последнего добавленного(после студента с заданной фамилией), если же студента с введенной пользователем фамилией не оказалось, то программа выводит сообщение об ошибке.