**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Одномерные статические массивы.

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 2372 |  | Котлова П. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение одномерных статических массивов; применение ввода, вывода, сортировка одномерных массивов; работа с условной инструкцией, инструкцией множественного выбора и циклами.

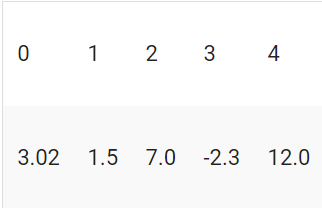
**Основные теоретические положения.**

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение.

Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

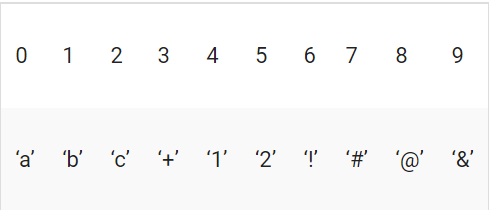
Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных.

Аналогом одномерного массива из математики может служить последовательность некоторых элементов с одним индексом: *ai*​ при i = 0, 1, 2, … n – одномерный вектор. Каждый элемент такой последовательности представляет собой некоторое значение определенного типа данных. Наглядно одномерный массив можно представить как набор пронумерованных ячеек, в каждой из которых содержится определенное значение:



Это пример одномерного массива из 5 элементов, каждый из которых представляет собой некоторое вещественное значение и каждое из этих значений имеет индекс от 0 до 4.

А вот пример одномерного массива из десяти элементов, представляющих собой одиночные символы:

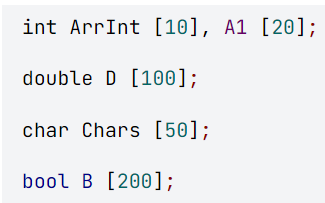


Каждый элемент в этих массивах определяется значением индекса элемента. Например, в последнем массиве элемент с индексом 5 равен символу ‘2’.

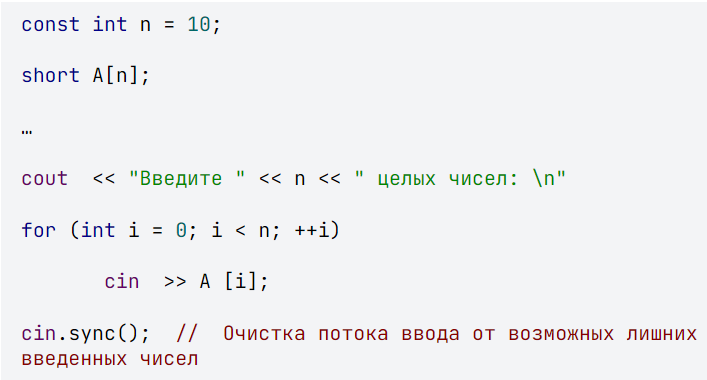
Объявление в программах одномерных массивов выполняется в соответствии со следующим правилом:

**<Базовый тип элементов> <Идентификатор массива> [<Количество элементов>]**

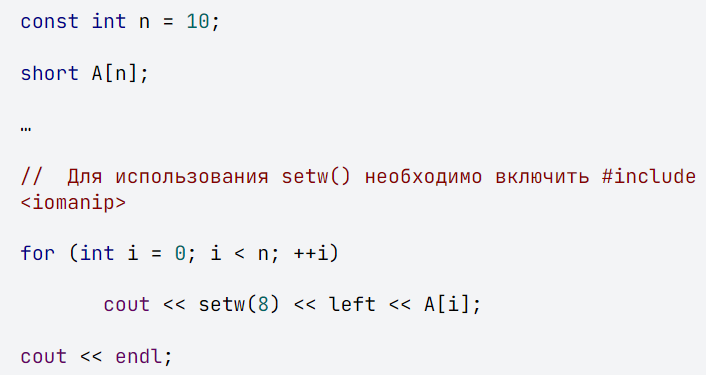
Например:



Простейший алгоритм ввода значений элементов одномерного массива может выглядеть так:



Простейший циклический алгоритм вывода значений элементов некоторого одномерного массива выглядит так:



**Постановка задачи.**

Необходимо написать программу, которая:

1)    Создает целочисленный массив размерности *N* = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.

2)    Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

3)    Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

4)    Выводит среднее значение (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Выводит индексы всех элементов, которые равны этому значению, и их количество. Подсчитайте время поиска.

5)    Выводит количество элементов в отсортированном массиве, которые меньше числа *a*, которое инициализируется пользователем.

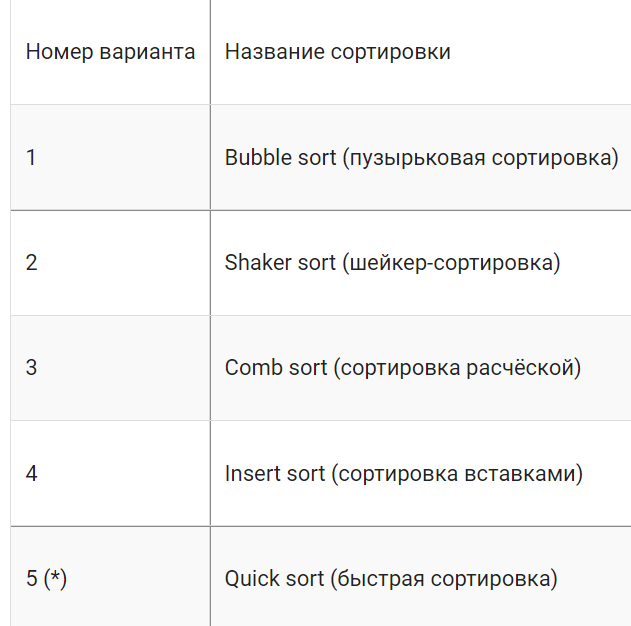
6)    Выводит количество элементов в отсортированном массиве, которые больше числа *b*, которое инициализируется пользователем.

7)    Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

8)     Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

Должна присутствовать возможность запуска каждого пункта многократно.

Таблица – Варианты сортировок

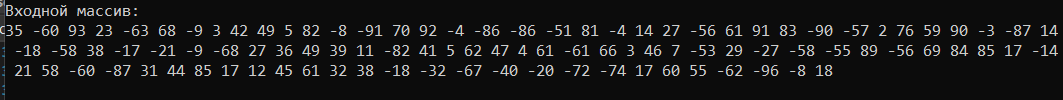


**Выполнение работы.**

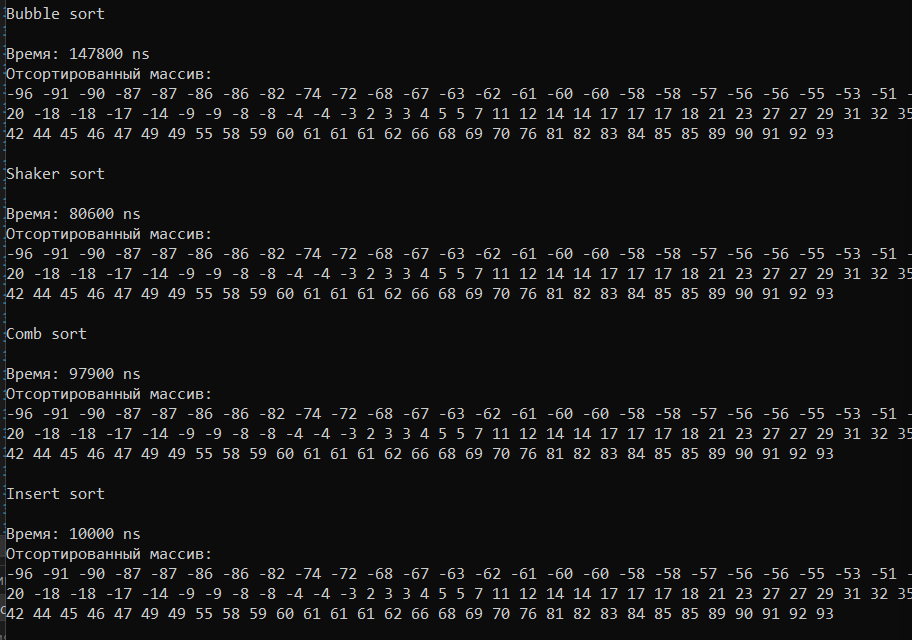
Код программы представлен в приложении А.

Описание кода и использованных алгоритмов.

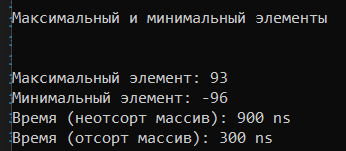
1. При запуске программы появляется окно, в котором вводится целочисленный массив размерности N = 100. Элементы массива принимают случайное значение в диапазоне от -99 до 99.



1. Дальше заданный массив выводится в отсортированном виде при помощи разных сортировок: Bubble sort, Shaker sort, Comb sort, Insert sort. Также выводится время, затраченное на каждую сортировку.

****

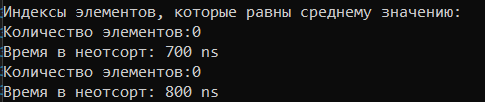
1. Затем выводятся максимальный и минимальный элементы массива, а также время поиска элементов в отсортированном массиве и неотсортированном.



1. Одновременно с максимальным и минимальным элементами появляется их среднее в отсортированном и неотсортированном массивах.

****

Дальше выводятся индексы всех элементов, которые равны этому значению, их количество и время поиска.

****

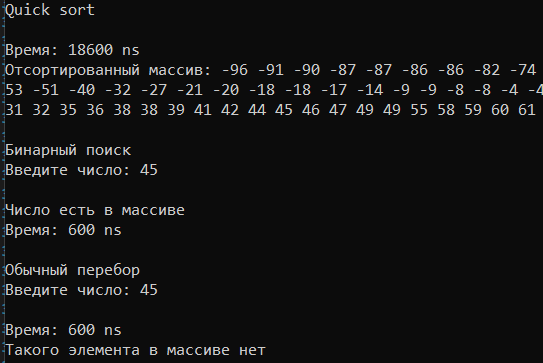
1. Затем пользователю предлагается ввести число. Выводятся количество элементов в отсортированном массиве, которые меньше этого числа.

****

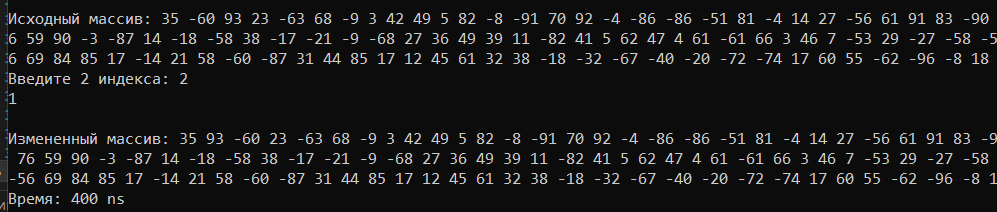
1. Пользователь снова вводит число. Программа выводит количество элементов в отсортированном массиве, которые больше заданного числа.

****

1. Выводится заданный в пункте 1 массив, отсортированный сортировкой Quick sort. После чего пользователь вводит число, которое программа находит с помощью алгоритма бинарного поиска и обычного перебора. Также программа выводит скорость работы двух алгоритмов.

****

1. Последним программа меняет местами элементы массива, индексы которых вводит пользователь. Выводятся два массива для сравнения (заданный в пункте 1 и измененный) и скорость обмена.

****

**Выводы.**

В ходе работы были изучены одномерные статические массивы, ввод, вывод, сортировка одномерных массивов и работа с условной инструкцией, инструкцией множественного выбора и циклами.

Приложение А

рабочий код

#include <chrono>

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

using namespace chrono;

void quicksort(int\* arr, int endn, int begin)

{

int mid;

int f = begin;

int l = endn;

mid = arr[(f + l) / 2];

while (f < l)

{

while (arr[f] < mid) f++;

while (arr[l] > mid) l--;

if (f <= l)

{

swap(arr[f], arr[l]);

f++;

l--;

}

}

if (begin < l) quicksort(arr, l, begin);

if (f < endn) quicksort(arr, endn, f);

}

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Russian");

auto start = steady\_clock::now();

cout << endl;

auto end = steady\_clock::now();

auto result = duration\_cast<nanoseconds>(end - start);

cout << "Время: " << result.count() << " ns";

cout << "\n\n";

cout << "Входной массив:\n";

const int n = 100;

int a[n], a\_sort[n];

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

{

a[i] = rand() % 199 - 99;

a\_sort[i] = a[i];

cout << a[i] << " ";

}

cout << "\n\nBubble sort";

cout << endl;

start = steady\_clock::now();

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

bool flag = true;

for (int j = 0; j < n - i - 1; j++)

if (a\_sort[j] > a\_sort[j + 1]) {

flag = false;

swap(a\_sort[j], a\_sort[j + 1]);

}

if (flag)

break;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя: " << result.count() << " ns";

cout << "\nОтсортированный массив:\n";

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

cout << a\_sort[i] << " ";

a\_sort[i] = a[i];

}

cout << "\n\nShaker sort";

cout << endl;

start = steady\_clock::now();

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

bool flag = true;

for (int j = 0; j < n - 1 - i; j++)

if (a\_sort[j] > a\_sort[j + 1]) {

flag = false;

swap(a\_sort[j], a\_sort[j + 1]);

}

for (int j = n - 2 - i; j > i; j--)

if (a\_sort[j] < a\_sort[j - 1]) {

flag = false;

swap(a\_sort[j], a\_sort[j - 1]);

}

if (flag)

break;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя: " << result.count() << " ns";

cout << "\nОтсортированный массив:\n";

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

cout << a\_sort[i] << " ";

a\_sort[i] = a[i];

}

cout << "\n\nComb sort";

cout << endl;

start = steady\_clock::now();

int count = 0;

float k = 1.247, S = n;

while (k < S)

{

for (int i = 0; i + k < S; i++)

{

if (a\_sort[i] > a\_sort[int(i + k)])

swap(a\_sort[i], a\_sort[int(i + k)]);

}

S /= k;

}

while (true)

{

for (int i = 0; i < n - 1; i++)

{

if (a\_sort[i] > a\_sort[i + 1])

{

swap(a\_sort[i], a\_sort[i + 1]);

}

else count++;

}

if (count == n - 1)

break;

else

count = 0;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя: " << result.count() << " ns";

cout << "\nОтсортированный массив:\n";

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

cout << a\_sort[i] << " ";

a\_sort[i] = a[i];

}

cout << "\n\nInsert sort";

cout << endl;

start = steady\_clock::now();

int x;

for (int i = 1; i < n; i++)

{

x = a\_sort[i];

int j = i;

while ((j > 0) && (x < a\_sort[j - 1]))

{

a\_sort[j] = a\_sort[j - 1];

j--;

}

a\_sort[j] = x;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя: " << result.count() << " ns";

cout << "\nОтсортированный массив:\n";

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

cout << a\_sort[i] << " ";

}

cout << "\n\nМаксимальный и минимальный элементы";

cout << endl;

// ищем max и min для неотсортированного массива

int max = a[0], min = a[0];

start = steady\_clock::now();

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

{

max = a[i] > max ? a[i] : max;

min = a[i] < min ? a[i] : min;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << endl;

cout << "\nМаксимальный элемент: " << max;

cout << "\nМинимальный элемент: " << min;

cout << "\nВремя (неотсорт массив): " << result.count() << " ns";

// ищем max и min для отсортированного массива

start = steady\_clock::now();

int max1 = a\_sort[0], min1 = a\_sort[99];

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя (отсорт массив): " << result.count() << " ns";

// сред значение max и min

float m, m1;

m = float((min + max) / 2);

cout << "\nСред значение max и min в неотсорт массиве: " << m;

m1 = float((min1 + max1) / 2);

cout << "\nСред значение max и min в отсорт массиве: " << m1;

//Выводит индексы всех элементов, которые равны этому значению, и их количество.

cout << "\n\nИндексы элементов, которые равны среднему значению:";

start = steady\_clock::now();

int count3 = 0;

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

if (a[i] == int(m)) {

cout << i << " ";

count3 += 1;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << endl << "Количество элементов:" << count3;

cout << "\nВремя в неотсорт: " << result.count() << " ns";

start = steady\_clock::now();

int count4 = 0;

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

if (a\_sort[i] == int(m)) {

cout << i << " ";

count4 += 1;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << endl << "Количество элементов:" << count4;

cout << "\nВремя в неотсорт: " << result.count() << " ns";

//Выводит количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.

cout << "\n\nВведите число a: ";

int an, count1 = 0;

cin >> an;

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

if (a\_sort[i] < an)

count1 += 1;

cout << "Количество элементов меньше а: " << count1;

//Выводит количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.

cout << "\nВведите число b: ";

int b, count2 = 0;

cin >> b;

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

if (a\_sort[i] > b)

count2 += 1;

cout << "Количество элементов больше b: " << count2;

cout << endl;

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

a\_sort[i] = a[i];

//\*

cout << "\n\nQuick sort";

cout << endl;

start = steady\_clock::now();

int endn = n - 1, begin = 0;

quicksort(a\_sort, endn, begin);

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя: " << result.count() << " ns";

cout << endl << "Отсортированный массив: ";

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

cout << a\_sort[i] << " ";

//Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве.

cout << "\n\nБинарный поиск";

cout << "\nВведите число: ";

int key;

cin >> key;

bool flag = false;

int l = 0;

int r = n - 1;

int mid;

start = steady\_clock::now();

while ((l <= r) && (flag != true))

{

mid = (l + r) / 2;

if (a\_sort[mid] == key)

flag = true;

if (a\_sort[mid] > key)

r = mid - 1;

else l = mid + 1;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

if (flag)

cout << "\nЧисло есть в массиве";

else

cout << "\nТакого элемента в массиве нет";

cout << "\nВремя: " << result.count() << " ns";

cout << "\n\nОбычный перебор";

cout << "\nВведите число: ";

int key1;

cin >> key1;

bool flag1 = false;

start = steady\_clock::now();

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

if (a[i] == key1)

flag1 = true;

else

flag1 = false;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\nВремя: " << result.count() << " ns";

if (flag1)

cout << "\nЧисло есть в массиве";

else

cout << "\nТакого элемента в массиве нет";

//Меняет местами элементы массива, индексы которых вводит пользователь.

cout << "\n\nИсходный массив: ";

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

cout << a[i] << " ";

cout << "\nВведите 2 индекса: ";

int c, d;

cin >> c >> d;

cout << "\nИзмененный массив: ";

start = steady\_clock::now();

int swap = a[c];

a[c] = a[d];

a[d] = swap;

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

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

cout << a[i] << " ";

}

cout << "\nВремя: " << result.count() << " ns\n"; //скорость обмена

system("pause");

return 0;

}

Условный пример содержания пункта «Выполеннеие работы»

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательности | Меню:    Проверка на ввод символов, которые не входят в диапазон выбора: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Создание новой записи | |
| При вводе пользователем корректного значения пункта меню и выбора создания новой записи, пользователь может добавить данные о студенте, но только на русском языке. | Для начала идёт проверка на правильный ввод пользователем:    Как только все данные заданы, пользователь может еще раз просмотреть все данные о студенте и в случае, если видит ошибку, сразу же ее исправить без дополнительного вызова меню: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Внесение изменений в имеющуюся запись | |
| Если пользователь допустил ошибку в создании карточки студента или данные изменились, он может внести изменения по пунктам, которые были созданы ранее. Это второе меню, с которым встречается пользователь. | При необходимости внесения изменений в запись о студенте, перед пользователем появляется меню:    Как только пользователь вводит корректное значение (в данном случае от 1 до 10), его ответ обрабатывается и предлагается изменить существующие данные на новые. После пользователь возвращается (при необходимости) в главное меню. |

Продолжение Таблицы

|  |  |
| --- | --- |
| Информация о студентах в количестве | |
| Если пользователь хочет классифицировать студентов по полу, по группам, по номеру в списке, то он может вывести список о всех студентах | Пример, когда пользователь выбрал список данных по конкретной группе, ввел номер группы, а студентов в этой группе пока что не имеется:    Пример, когда пользователь выбрал студентов по их номеру в списке соответственно для двух групп: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Рейтинг студентов | |
| Пользователь в главном меню может выбрать такую опцию как рейтинг студентов, чтобы наглядно увидеть список студентов с минимальными данными и рейтингу их оценок. | Выводится номер в списке по группам, ФИО, сама группа и рейтинг. Рейтинг складывается из всех оценок за сессию (диф зачеты экзамены):    В данном примере студентов в базе было занесено один, то есть, информация и средний бал вывелся только по одному человеку. |

Продолжение Таблицы

|  |  |
| --- | --- |
| Стипендия | |
| Стипендию пользователь может узнать в двух случаях: он может выбрать непосредственно пункт в меню, где ему выйдет список студентов, которые будут получать стипендию. Или пользователь может выбрать список с данными о студентах, которые могли бы получать стипендию, то есть они учатся не на очной форме обучения при оценках не ниже хорошо. | Пример вывода данных о студенте, который не будет получать стипендию при положительных оценках: |

Окончание Таблицы

|  |  |
| --- | --- |
| Очередь в общежитие | |
| Пользователь может узнать очередь в общежитие, занося данные о место происхождении студента и о его заработке. | Пример очереди из студентов, которые получают разных доход (в тыс. руб.), являются и иногородними, и местными:  ../Desktop/очередь.png |
| Полный список о студентах | |
| Если пользователь не собирается получать данные о студентах по конкретному критерию, он может вывести полный список студентов в консоли или же получить отдельный файл. | Вывод в консоли:  ../Desktop/Снимок%20экрана%202020-02-29%20в%2010.22.33.png  Вывод в файл:  ../Desktop/Снимок%20экрана%202020-02-29%20в%2010.22.07.png |