**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра «АПУ»**

**ОТЧЁТ**

**по лабораторной работе № 5**

**по дисциплине «Программирование»**

**«Пользовательские типы данных»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 4391 | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | Мухачев Д. О. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Писарев А.С. |

Санкт-Петербург

2024

**Цель работы:**

Задачей было ввести данные определенное количество студентов и выполнить операции фильтрации, группировки, агрегации и сортировки с ними.

**Ход работы:**

Первым шагом было создания нового типа данных – структуры, содержащий ФИО, номер группы и оценки студента:

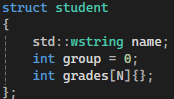
****

Рисунок 1

Вторым шагом – функции сортировки массивов по имени, номеру группы и среднему баллу:

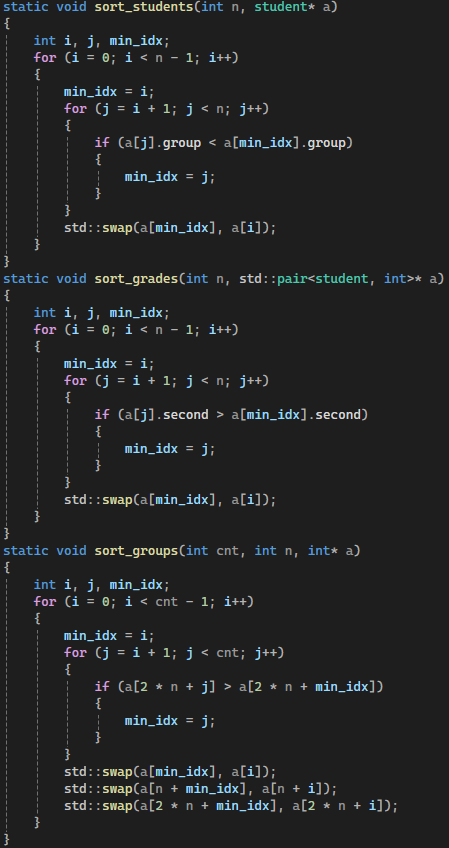


Рисунок 2

Далее был написан основной цикл программы, включающий в себя интерфейс, позволяющий вводить данные студентов в соответствии с условиями задачи и использующий Unicode для ввода/вывода кириллицы:

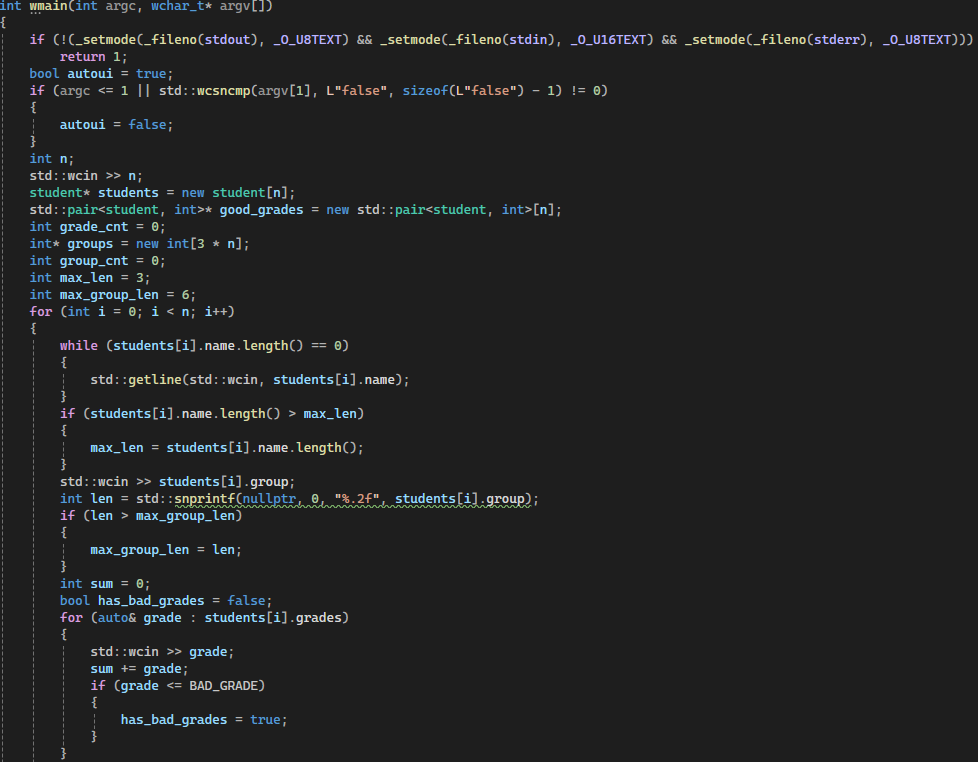


Рисунок 3

Программа группирует студентов по их группам, ведет подсчет двоечников в каждой группе, считает средний балл каждого студента:

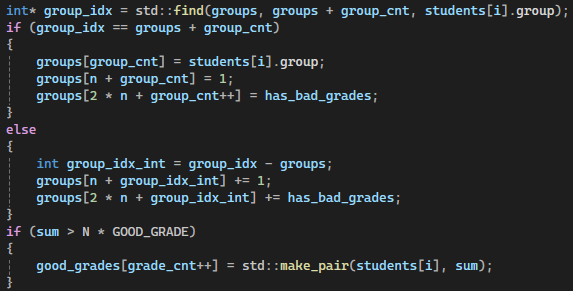


Рисунок 4

После завершения основного цикла программы массивы сортируются с использованием соответствующий функций:



Рисунок 5

Программа выводит отсортированный по ФИО массив студентов в формате, требуемом в условии задачи:

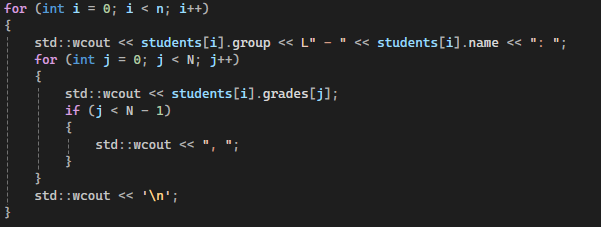


Рисунок 6

Потом выводятся массив всех студентов с высокой успеваемостью, отсортированный по среднему баллу и массив всех групп, отсортированный по количеству двоечников, затем высвобождает память и звершает работу:

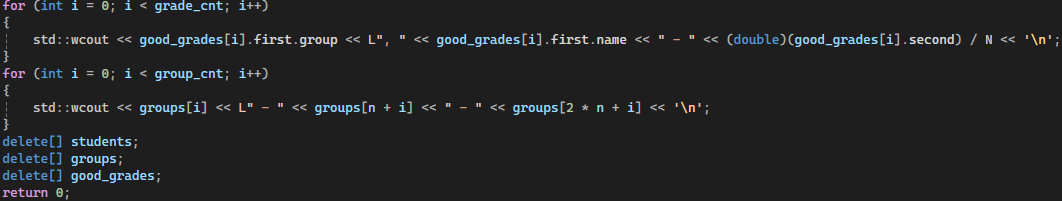


Рисунок 7

В завершении, были подключены все необходимые для работы программы библиотеки:

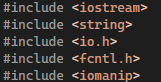


Рисунок 8

**Проверка работоспособности программы:**

Для теста были использованы входные данные из методических материалов:



Рисунок 11

Полученные результаты:

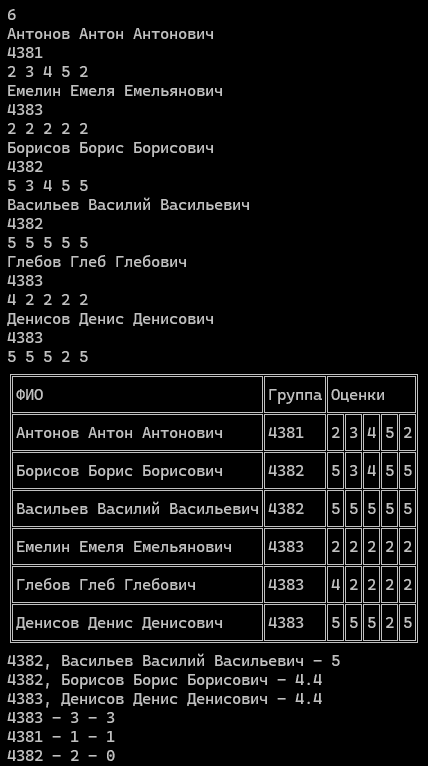


Рисунок 12

**Вывод:** В ходе выполнения лабораторной работы, целью которой была разработка программы для выполнения некоторых действий над пользовательскими типами данных, был получен практический опыт в нескольких ключевых областях. Было освоено создание различных типов интерфейсов, использование циклов, условных операторов и разработка функций для решения математических задач. Теоретические знания, полученные на лекциях, помогли в написании алгоритмов сортировки.

Приложение 1

Код программы

#include <iostream>

#include <string>

#include <io.h>

#include <fcntl.h>

#include <iomanip>

constexpr int N = 5, GOOD\_GRADE = 4, BAD\_GRADE = 2;

constexpr wchar\_t LINE\_RL = L'\u2550';

constexpr wchar\_t LINE\_TB = L'\u2551';

constexpr wchar\_t LINE\_TR = L'\u255A';

constexpr wchar\_t LINE\_TL = L'\u255D';

constexpr wchar\_t LINE\_BR = L'\u2554';

constexpr wchar\_t LINE\_BL = L'\u2557';

constexpr wchar\_t LINE\_TBR = L'\u2560';

constexpr wchar\_t LINE\_TBL = L'\u2563';

constexpr wchar\_t LINE\_BRL = L'\u2566';

constexpr wchar\_t LINE\_TRL = L'\u2569';

constexpr wchar\_t LINE\_TBRL = L'\u256C';

static void out\_sep(int W1, int W2, int W3, bool begin)

{

int step = W3 / (N - 1);

std::wcout << '\n' << LINE\_TBR;

for (int i = 0; i < W1; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_TBRL;

for (int i = 0; i < W2; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_TBRL;

for (int i = 0; i < W3; i++)

{

if (i % step == 1)

{

if (begin)

{

std::wcout << LINE\_BRL;

}

else

{

std::wcout << LINE\_TBRL;

}

}

else

{

std::wcout << LINE\_RL;

}

}

std::wcout << LINE\_TBL << '\n';

}

static void out\_array(int ALIGN\_W, int LEN, const char\* const name, double\* a)

{

std::wcout << LINE\_TBL << '\n' << LINE\_TB;

std::wcout << std::left << std::setw(ALIGN\_W) << std::setfill(L' ') << name;

std::wcout << LINE\_TB;

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

{

std::wcout << std::right << std::setw(ALIGN\_W) << std::setfill(L' ') << a[i];

std::wcout << LINE\_TB;

}

}

static void out\_begin(int ALIGN\_W, int LEN, int n, const char\* const\* const names, double\*\* arrs)

{

std::wcout << LINE\_TR;

for (int i = 0; i < LEN + 1; i++)

{

for (int j = 0; j < ALIGN\_W; j++)

{

std::wcout << LINE\_RL;

}

if (i < LEN)

{

std::wcout << LINE\_BRL;

}

}

std::wcout << LINE\_TL << '\n' << LINE\_TB;

std::wcout << std::left << std::setw(ALIGN\_W) << std::setfill(L' ') << "idx";

}

static void out\_end(int W1, int W2, int W3)

{

int step = W3 / (N - 1);

std::wcout << '\n' << LINE\_TR;

for (int i = 0; i < W1; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_TRL;

for (int i = 0; i < W2; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_TRL;

for (int i = 0; i < W3; i++)

{

if (i % step == 1)

{

std::wcout << LINE\_TRL;

}

else

{

std::wcout << LINE\_RL;

}

}

std::wcout << LINE\_TL << '\n';

}

struct student

{

std::wstring name;

int group = 0;

int grades[N]{};

};

static void sort\_students(int n, student\* a)

{

int i, j, min\_idx;

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

{

min\_idx = i;

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

{

if (a[j].group < a[min\_idx].group)

{

min\_idx = j;

}

}

std::swap(a[min\_idx], a[i]);

}

}

static void sort\_grades(int n, std::pair<student, int>\* a)

{

int i, j, min\_idx;

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

{

min\_idx = i;

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

{

if (a[j].second > a[min\_idx].second)

{

min\_idx = j;

}

}

std::swap(a[min\_idx], a[i]);

}

}

static void sort\_groups(int cnt, int n, int\* a)

{

int i, j, min\_idx;

for (i = 0; i < cnt - 1; i++)

{

min\_idx = i;

for (j = i + 1; j < cnt; j++)

{

if (a[2 \* n + j] > a[2 \* n + min\_idx])

{

min\_idx = j;

}

}

std::swap(a[min\_idx], a[i]);

std::swap(a[n + min\_idx], a[n + i]);

std::swap(a[2 \* n + min\_idx], a[2 \* n + i]);

}

}

int wmain(int argc, wchar\_t\* argv[])

{

if (!(\_setmode(\_fileno(stdout), \_O\_U8TEXT) && \_setmode(\_fileno(stdin), \_O\_U16TEXT) && \_setmode(\_fileno(stderr), \_O\_U8TEXT)))

return 1;

bool autoui = true;

if (argc <= 1 || std::wcsncmp(argv[1], L"false", sizeof(L"false") - 1) != 0)

{

autoui = false;

}

int n;

std::wcin >> n;

student\* students = new student[n];

std::pair<student, int>\* good\_grades = new std::pair<student, int>[n];

int grade\_cnt = 0;

int\* groups = new int[3 \* n];

int group\_cnt = 0;

int max\_len = 3;

int max\_group\_len = 6;

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

{

while (students[i].name.length() == 0)

{

std::getline(std::wcin, students[i].name);

}

if (students[i].name.length() > max\_len)

{

max\_len = students[i].name.length();

}

std::wcin >> students[i].group;

int len = std::snprintf(nullptr, 0, "%.2f", students[i].group);

if (len > max\_group\_len)

{

max\_group\_len = len;

}

int sum = 0;

bool has\_bad\_grades = false;

for (auto& grade : students[i].grades)

{

std::wcin >> grade;

sum += grade;

if (grade <= BAD\_GRADE)

{

has\_bad\_grades = true;

}

}

int\* group\_idx = std::find(groups, groups + group\_cnt, students[i].group);

if (group\_idx == groups + group\_cnt)

{

groups[group\_cnt] = students[i].group;

groups[n + group\_cnt] = 1;

groups[2 \* n + group\_cnt++] = has\_bad\_grades;

}

else

{

int group\_idx\_int = group\_idx - groups;

groups[n + group\_idx\_int] += 1;

groups[2 \* n + group\_idx\_int] += has\_bad\_grades;

}

if (sum > N \* GOOD\_GRADE)

{

good\_grades[grade\_cnt++] = std::make\_pair(students[i], sum);

}

}

sort\_students(n, students);

sort\_grades(grade\_cnt, good\_grades);

sort\_groups(group\_cnt, n, groups);

if (autoui)

{

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

{

std::wcout << students[i].group << L" - " << students[i].name << ": ";

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

{

std::wcout << students[i].grades[j];

if (j < N - 1)

{

std::wcout << ", ";

}

}

std::wcout << '\n';

}

}

else

{

std::wcout << LINE\_BR;

for (int i = 0; i < max\_len; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_BRL;

for (int i = 0; i < max\_group\_len; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_BRL;

for (int i = 0; i < 2 \* N - 1; i++)

{

std::wcout << LINE\_RL;

}

std::wcout << LINE\_BL << '\n' << LINE\_TB;

std::wcout << std::left << std::setw(max\_len) << std::setfill(L' ') << L"ФИО";

std::wcout << LINE\_TB;

std::wcout << std::left << std::setw(max\_group\_len) << std::setfill(L' ') << L"Группа";

std::wcout << LINE\_TB;

std::wcout << std::left << std::setw(2 \* N - 1) << std::setfill(L' ') << L"Оценки";

std::wcout << LINE\_TB;

out\_sep(max\_len, max\_group\_len, 2 \* N - 1, true);

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

{

std::wcout << LINE\_TB;

std::wcout << std::left << std::setw(max\_len) << std::setfill(L' ') << students[i].name;

std::wcout << LINE\_TB;

std::wcout << std::left << std::setw(max\_group\_len) << std::setfill(L' ') << students[i].group;

std::wcout << LINE\_TB;

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

{

std::wcout << students[i].grades[j] << LINE\_TB;

}

if (i < n - 1)

{

out\_sep(max\_len, max\_group\_len, 2 \* N - 1, false);

}

}

out\_end(max\_len, max\_group\_len, 2 \* N - 1);

}

for (int i = 0; i < grade\_cnt; i++)

{

std::wcout << good\_grades[i].first.group << L", " << good\_grades[i].first.name << " - " << (double)(good\_grades[i].second) / N << '\n';

}

for (int i = 0; i < group\_cnt; i++)

{

std::wcout << groups[i] << L" - " << groups[n + i] << " - " << groups[2 \* n + i] << '\n';

}

delete[] students;

delete[] groups;

delete[] good\_grades;

return 0;

}