МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХ™НИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ПРАКТИЧЕСКАЯ РАБОТА №3**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-323», «АВТФ»  *Раков Михаил Андреевич*  «2» июня 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Архипова Анастасия Борисовна*  «\_\_\_» июня 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2024

**Цели и задачи работы:** изучение функционала библиотек algorithm и iterator на примере собственных задач.

**Задание к работе:** придумать и реализовать 10 задач с использованием алгоритмов и итераторов.

1. **Задание №1.**

Преподаватель расставил баллы за семестр, однако на сайте университета произошёл сбой. Некоторые студенты были записаны в базу данных дважды. Для экономии времени, было решено оставить записи с наибольшими баллами.

Дан список студентов с их итоговыми баллами, удалите повторы с меньшим количеством баллов, после чего выведите студентов закончивших семестр на 5.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <vector>

using namespace std;

struct Student

{

string name;

int score;

};

int main()

{

vector<Student> students = {

{"Александр", 95},

{"Ирина", 85},

{"Дмитрий", 90},

{"Елена", 75},

{"Оксана", 100},

{"Даниил", 80},

{"Дмитрий", 64}

};

sort(students.begin(), students.end(), [](const Student& a, const Student& b) {

return a.name < b.name;

});

auto endU = unique(students.begin(), students.end(), [](const Student& a, const Student& b) {

return a.name == b.name;

});

students.erase(endU, students.end());

vector<Student> excellent\_students;

copy\_if(students.begin(), students.end(), back\_inserter(excellent\_students), [](const Student& student) {

return student.score >= 90;

});

cout << "Студенты, сдавшие экзамен на отлично:" << endl;

for (const auto& [name, score] : excellent\_students)

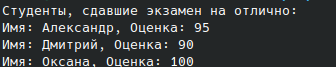
{

cout << "Имя: " << name << ", Оценка: " << score << endl;

}

}

**Результат работы программы**



1. **Задание №2.**

В 29 школе постепенно переходят от к бумажного журнала к электронному. Некоторые учителя забывают переносить оценки из одного в другой. В конце года журналы попытались объединить, но сделали это неправильно. Вместо того, чтобы добавить оценки к уже существующим предметам, в таблице были созданы новые предметы, повторяющие названия уже существующих.

Необходимо исправить сложившуюся ситуацию: объедение оценки одинаковых предметов и удалите повторы, после чего посчитайте средний балл по каждому предмету.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <numeric>

#include <vector>

using namespace std;

template<typename ForwardIt, typename Predicate>

ForwardIt my\_unique (ForwardIt first, ForwardIt last, Predicate pred)

{

first = adjacent\_find(first, last, pred);

if (first == last)

return last;

ForwardIt dest = first;

++first;

while (++first != last)

{

if (not pred(\*dest, \*first))

{

iter\_swap(++dest, first);

}

}

return ++dest;

}

struct Subject

{

string name;

vector<int> guards;

bool operator< (const Subject& rhs) const

{

return this->name < rhs.name;

}

bool operator== (const Subject& rhs) const

{

return this->name == rhs.name;

}

};

int main()

{

vector<Subject> subjects = {

{"русский язык", {4, 5, 3, 4, 4}},

{"алгебра", {5, 4, 3}},

{"история", {3, 4, 2, 4}},

{"физика", {4, 5, 5, 5, 4, 5}},

{"история", {5}},

{"химия", {4, 5, 4, 4, 4, 4, 3}},

{"алгебра", {5, 4, 5}}

};

sort(subjects.begin(), subjects.end());

auto endU = my\_unique(subjects.begin(), subjects.end(), [](const Subject& a, const Subject& b) {

return a.name == b.name;

});

for (auto itDbl = endU; itDbl != subjects.end(); ++itDbl)

{

auto itU = find (subjects.begin(), endU, \*itDbl);

copy(itDbl->guards.begin(), itDbl->guards.end(), back\_inserter(itU->guards));

}

subjects.erase(endU, subjects.end());

for (auto const& [name, guards] : subjects)

{

cout << name << ": ";

for (auto el : guards)

{

cout << el << " ";

}

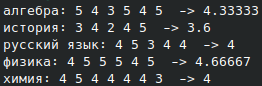
double sumG = accumulate(guards.begin(), guards.end(), 0.0);

cout << " -> " << (sumG / guards.size()) << endl;

}

}

**Результат работы программы**



1. **Задание №3.**

Найдите и выведите все простые числа от 2 до n, используя решето Эратосфена.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <numeric>

#include <vector>

using namespace std;

int main()

{

size\_t n = 100;

vector<int> nums(n);

iota(nums.begin(), nums.end(), 2);

auto endPrimes = nums.end();

for (auto it = nums.begin(); it != endPrimes; ++it)

{

endPrimes = remove\_if(next(it), endPrimes, [it] (int a) {

return a % \*it == 0;

});

}

nums.erase(endPrimes, nums.end());

cout << "Простые числа от 2 до " << (n+2) << ":" << endl;

for (auto el : nums)

{

cout << el << " ";

}

cout << end

**Результат работы программы**



1. **Задание №4.**

На соревнованиях по фигурному катанию, оценки участникам рассчитываются по следующему принципу. Каждый из судей выставляет свою оценку от 0.0 до 6.0, после чего из списка удаляются максимальные и минимальные баллы. Итоговая оценка равна среднему арифметическому оставшихся баллов. Дан список оценок участника, выведите баллы без учета максимальных минимальных и итоговый балл.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <numeric>

#include <vector>

using namespace std;

int main()

{

vector<double> points = {5.5, 6.0, 5.0, 5.2, 6.0, 5.1, 5.4, 5.8};

auto [minIt, maxIt] = minmax\_element(points.begin(), points.end());

double minP = \*minIt, maxP = \*maxIt;

auto newEnd = remove(points.begin(), points.end(), minP);

newEnd = remove(points.begin(), newEnd, maxP);

points.erase(newEnd, points.end());

double sumPoints = accumulate(points.begin(), points.end(), 0.0);

cout << "Баллы без учета максимальных и минимальных: " << endl;

for (auto el : points)

{

cout << el << " ";

}

cout << endl << "Средний балл: " << (sumPoints / points.size()) << endl;

}

**Результат работы программы**



1. **Задание №5.**

Необходимо найти количество полных нечётных квадратов среди чисел от 0 до n.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <cmath>

#include <numeric>

#include <vector>

using namespace std;

bool is\_not\_even (int a)

{

return a % 2 != 0;

}

bool is\_sqr (int a)

{

int \_sqrt = sqrt(a);

return a == pow(\_sqrt, 2);

}

int main()

{

size\_t n = 100;

vector<int> nums(n);

iota(nums.begin(), nums.end(), 0);

auto it = partition(nums.begin(), nums.end(), is\_not\_even);

cout << "Количество полных нечётных квадратов от 0 до " << n

<< " = " << count\_if (nums.begin(), it, is\_sqr)

<< endl;

}

**Результат работы программы**



1. **Задание №6.**

Почтовый адрес сотрудника НГТУ составляется по следующему образцу**:** *[фамилия](mailto:фамилия@corp.nstu.ru)*[@corp.nstu.ru; почтовый](mailto:фамилия@corp.nstu.ru) адрес студентов: *[фамилия.год поступления](mailto:фамилия@corp.nstu.ru)*[@stud.nstu.ru.](mailto:фамилия@corp.nstu.ru) Дан список почтовых адресов, необходимо найти и вывести те из них, что принадлежат сотрудникам университета.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <cmath>

#include <numeric>

#include <vector>

using namespace std;

bool str\_search (string const& str, string const& s)

{

auto itS = search(str.begin(), str.end(), s.begin(), s.end());

return itS != str.end();

}

int main()

{

vector<string> mailAddresses = {

"seleznev@corp.nstu.ru",

"rakov.2023@stud.nstu.ru",

"avdeenko@corp.nstu.ru",

"levich.2021@stud.nstu.ru",

"ryabova.2022@stud.nstu.ru",

"cosarev@corp.nstu.ru",

"ryazanov.2023@stud.nstu.ru",

"aksenov.2023@stud.nstu.ru"

};

vector<string> corpAddresses;

string filterWord = "corp";

copy\_if(mailAddresses.begin(), mailAddresses.end(), back\_inserter(corpAddresses),

[&filterWord] (string const& str){

return str\_search(str, filterWord);

});

for (auto const& s : corpAddresses)

{

cout << s << endl;

}

}

**Результат работы программы**



1. **Задание №7.**

Перед выходом фильма в прокат, режиссёру необходимо удалить некоторые сцены из фильма. При этом фильм должен остаться полнометражным (его длительность - 52 минуты и более). Режиссёр хочет удалить либо самую продолжительную сцену, из тех что возможно удалить, либо как можно большее количество коротких сцен.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <cmath>

#include <numeric>

#include <vector>

using namespace std;

int main()

{

std::vector<int> scenesDur {

5, 10, 7, 12, 9, 22, 4

};

int minDurFilm = 52;

int durationFilm = accumulate(scenesDur.begin(), scenesDur.end(), 0);

int diffDur = durationFilm - minDurFilm;

int criticDur = 13;

sort(scenesDur.begin(), scenesDur.end());

auto maxDurScene = upper\_bound(scenesDur.begin(), scenesDur.end(), diffDur);

maxDurScene = prev(maxDurScene);

vector<int> durationSum;

partial\_sum(scenesDur.begin(), scenesDur.end(), back\_inserter(durationSum));

auto it = upper\_bound(durationSum.begin(), durationSum.end(), diffDur);

int mxCntScene = distance(durationSum.begin(), it);

int durShortScenes = \*prev(it);

cout << "Наибольшая продолжительность сцены, которую можно вырезать: "

<< \*maxDurScene << endl;

cout << "Количество коротких сцен, которые можно вырезать: "

<< mxCntScene << ", их продолжительность: "

<< durShortScenes << endl;

}

**Результат работы программы**



1. **Задание №8.**

Андрей решил продать некоторые свои старые вещи в ломбард. Ломбард принимает по две вещи за раз и выплачивает сумму, равную разности цен вещей. Дан список примерной стоимости вещей Андрея. Помогите ему заработать как можно больше.

**Реализация на C++**

#include <algorithm>

#include <iostream>

#include <vector>

using namespace std;

int main()

{

std::vector<int> prices {700, 1500, 565, 300, 623, 400};

int cntThings = 3;

int maxProfit = 0;

while (cntThings--)

{

auto maxPrice = max\_element(prices.begin(), prices.end());

int profit = \*maxPrice;

prices.erase(maxPrice);

auto minPrice = min\_element(prices.begin(), prices.end());

profit -= \*minPrice;

prices.erase(minPrice);

maxProfit += profit;

}

cout << "Максимальная прибыль: " << maxProfit << endl;

}

**Результат работы программы**



1. **Задание №9.**

Вы хотите купить фрукты. Есть два магазина рядом с вашим домом, и вы знаете, какие фрукты продаются в каждом из них. Вам нужно определить, какие фрукты продаются в обоих магазинах, и указать, в каком магазине их лучше купить по более низкой цене.

**Реализация на C++**

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

struct Product

{

string name;

double price;

};

bool cmp\_names (const Product& p1, const Product& p2)

{

return p1.name < p2.name;

}

int main()

{

vector<Product> store1 =

{

{"Грейпфрут", 119.99},

{"Яблоки", 140},

{"Бананы", 140},

{"Груши", 190.75},

{"Апельсины", 125}

};

vector<Product> store2 =

{

{"Груши", 169.50},

{"Мандарины", 144},

{"Апельсины", 139.99},

{"Манго", 370},

{"Яблоки", 155.30}

};

sort (store1.begin(), store1.end(), cmp\_names);

sort (store2.begin(), store2.end(), cmp\_names);

vector<Product> intersection;

set\_intersection(store1.begin(), store1.end(), store2.begin(), store2.end(),

back\_inserter(intersection), cmp\_names);

for (const auto& product : intersection)

{

auto pred = [&product] (const Product& p) {

return product.name == p.name;

};

double p1 = find\_if(store1.begin(), store1.end(), pred) -> price;

double p2 = find\_if(store2.begin(), store2.end(), pred) -> price;

cout << product.name << " лучше купить"

<< (p1 < p2 ? " в первом" : " во втором") << " магазине"

<< " по " << min(p1, p2) << " руб. за кг" << endl;

}

}

**Результат работы программы**



1. **Задание №10.**

Дан список целых чисел. Необходимо найти количество последовательных пар, в которых первый элемент меньше второго, количество элементов кратных трём, количество пар сумма элементов которых равна 10.

**Реализация на C++**

#include <iostream>

#include <vector>

#include <algorithm>

#include <map>

using namespace std;

int main()

{

vector<int> numbers;

int number;

cout << "Введите массив целых чисел: ";

while (cin >> number)

{

numbers.push\_back(number);

}

int cntGreater = 0;

for (auto it = numbers.begin() + 1; it != numbers.end(); ++it)

{

if (\*it > \*prev(it))

{

++cntGreater;

}

}

int cntDiv3 = count\_if(numbers.begin(), numbers.end(), [] (int n) {

return n % 3 == 0;

});

int cntSum10 = 0;

for (auto it1 = numbers.begin(); it1 != numbers.end(); ++it1)

{

for (auto it2 = next(it1); it2 != numbers.end(); ++it2)

{

if (\*it1 + \*it2 == 10)

{

++cntSum10;

}

}

}

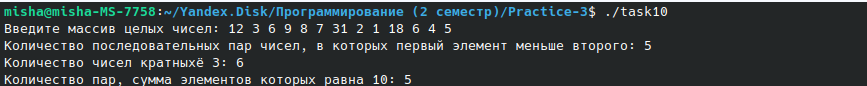
cout << "Количество последовательных пар чисел, в которых первый элемент меньше второго: " << cntGreater << endl;

cout << "Количество чисел кратных 3: " << cntDiv3 << endl;

cout << "Количество пар, сумма элементов которых равна 10: " << cntSum10 << endl;

}

**Результат работы программы**



**Вывод**

Таким образом, у меня получилось реализовать задачи с использованием алгоритмов и итераторов.