1. **Имеем чистый C. Напишите generic функцию линейного поиска в массиве. И приведите пример использования вашей функции.**

void \*linear\_search(void \*array, const size\_t count, const size\_t element\_size, const void \*target, size\_t \*output\_index) {

unsigned char \*ptr = array;

for (size\_t i = 0; i < count; i++) {

if (memcmp(ptr + i \* element\_size, target, element\_size) == 0) {

if (output\_index) {

\*output\_index = i;

}

return ptr + i \* element\_size;

}

}

if (output\_index) {

\*output\_index = INDEX\_NOT\_FOUND;

}

return NULL;

}

В функцию передается указатель void на начало данных, количество элементов, размер элемента, то, что ищем, выходной индекс. Сравнение идет побайтовое, если мы нашли, то выходим из функции, если нет, то индекс не найден. Можно было бы еще написать отдельную функцию компаратор для каждого типа как мы захотим и передавать ее в линейный поиск, так было бы гибче, но дольше.

Еще я написал функцию с использованием intrinsic AVX512 для int 32 которая в большинстве случаев быстрее стандартной функции std::ranges::find, но это очень платформозависимая оптимизация, даже в пределах одной архитектуры и поколений процессоров, надо проделать огромное количество тестирований на массивах разной размерности, с разной позицией искомого элемента. Во многих клиентских процессорах стоит сильный даунгрейд на использование AVX512 по причине перегрева процессора и специальные механизмы снижения тактовой частоты, упор в пропускную способность кэшей, переключения между скалярными режимами работы и смешиванием наборов avx с sse и тд. Я постарался подробно расписать что там происходит в extra\_find\_avx512.cpp, но можно пробовать еще много разных подходов с анроллом, префетчем, обработкой хвоста с масочными регистрами k0-k7 и предкомпилированной таблицей. Тестировал на миллионах элементов и миллиардах, прирост +- одинаковый. Можно использовать Intel VTune Profiler для просмотра производительности, памяти, там все подробно расписано по сотням метрик процессора, но это уже как будто тема какой-то статьи научной получится) Вот один из примеров прогона:



A screenshot of a computer

AI-generated content may be incorrect.

