

Лабораторная работа №1. Исследование особенностей реализации классического алгоритма умножения матриц

1.1 Цель лабораторной работы

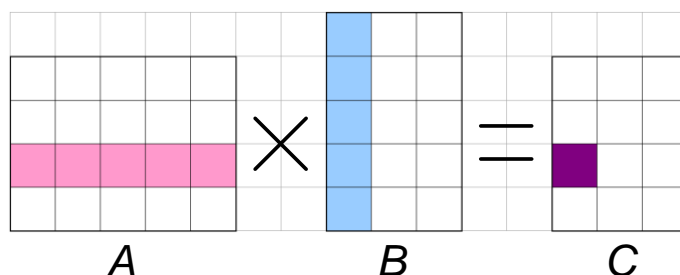
Рассмотреть различные способы хранения матриц, исследовать их умножение.

1.2. Теоретический материал

1.2.1 Оптимизации в задаче умножения матриц

Произведением матрицы A размера $N \times K$ и матрицы B размера $K \times M$ называется матрица C размера $N \times M$, элементы которой равны скалярному произведению соответствующих строки матрицы A и столбца матрицы B

$$c_{ij} = \sum_{k=1}^K a_{ik} \cdot b_{kj}.$$



Умножение матриц реализуется при помощи трех вложенных циклов. Оптимизировать данный алгоритм можно, изменяя порядок циклов. Одна из главных причин низкой производительности при «неправильном порядке» циклов – плохо организованная работа с памятью.

Современные процессоры чувствительны к тому, в каком порядке происходит чтение и запись в память. Чувствительность связана со сложной архитектурой организации памяти, как процессоров, так и сопроцессоров. Если чтение и запись происходит последовательно, то процессор может это предсказать и заранее загрузить данные в КЭШ-память. Доступ к кэш-памяти гораздо быстрее доступа к оперативной памяти. Кроме того, данные в кэш-память загружаются не поэлементно, а сразу группами (читаются данные в размере КЭШ-линии). Если из КЭШ-линии использовать только один элемент, то много данных будет загружено

в процессор и не использовано в вычислениях. Таким образом, увеличивается процент КЭШ-промахов, а значит и время работы программы.

Второй способ оптимизации вычислений – использование одномерных массивов для хранения матриц. Это позволяет делать в два раза меньше обращений к памяти (у двумерных массивов сначала происходит обращение к массиву указателей, а потом к самому элементу).

1.2.2 Замер времени выполнения программы в языке C++

Для измерения интервалов времени в языке C++ начиная с C++11 можно использовать функции и классы, содержащиеся в заголовочном файле **chrono**.

Например, приведенный ниже код замеряет время суммирования элементов массива *arr* размера *n*.

```
std::chrono::time_point<std::chrono::high_resolution_clock> start,
end;
start = std::chrono::high_resolution_clock::now();
for (int i = 0; i < n; ++i)
    sum = sum + arr[i];
end = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> diff = end - start;
std::cout << "time to sum = " << diff.count() << "s\n";
```

Для замера процессорного времени необходимо использовать функции, которые могут различаться для разных операционных систем.

1.3. Задание на лабораторную работу

1. Создать две целочисленные матрицы размера $N \times N$ с использованием двумерных динамических массивов. Реализовать классический алгоритм умножения матриц.

Провести тестирование программы на матрицах размерности $N = 512, 1024$ и 2048 без оптимизации (с ключом `-O0` или `-Od`, в конфигурации Debug в MS Visual Studio) и с ключом оптимизации `-O2` (кофигурация Release в MS Visual Studio). На каждом примере запустить не менее 3 раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сделать выводы.

*Таблица 1 – Время выполнения алгоритма умножения матриц
с двойными указателями, с*

Размер матрицы	Ключ оптимизации	
	-O0	-O2
512		
1024		
2048		

2. Создать две целочисленные матрицы размера $N \times N$ с использованием одномерных динамических массивов. Реализовать классический алгоритм умножения матриц.

Провести тестирование программы на матрицах размерности $N = 512, 1024$ и 2048 с ключом оптимизации `-O2` (конфигурация Release в MS Visual Studio). На каждом примере запустить не менее 3 раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить полученные результаты. Сделать выводы.

*Таблица 2 – Время выполнения алгоритма умножения матриц
с одинарными и двойными указателями, с*

Размер матрицы	Способ хранения матрицы в памяти	
	двойные указатели	одинарные указатели
512		
1024		
2048		

3. Для матриц, хранящихся в одномерных динамических массивах, реализовать классический алгоритм умножения со всеми возможными перестановками порядка циклов.

Провести тестирование программ на матрицах размерности $N = 512, 1024$ и 2048 . На каждом примере запустить не менее 3 раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты. Сделать выводы.

Таблица 3 – Время выполнения классического алгоритма умножения матриц

Размерность	Порядок циклов					
	ijk	ikj	jik	jki	kij	kji
512						
1024						
2048						

4. Выполнить задания 1-2 с использованием представления матриц в виде одномерных динамических массивов.

1.4. Результаты лабораторной работы

Результаты лабораторной работы представляются в виде отчета по лабораторной работе. В отчет включается титульный лист, цель работы, задание на лабораторную работу, листинг с комментариями, полученные результаты и выводы по лабораторной работе.

Отчет оформляется в электронном виде и высылается на e-mail vbyzov.vyatsu@gmail.com (в теме или тексте письма, а также в названии документа с отчетом должны фигурировать ФИ студента, его группа, номер лабораторной работы).

Лабораторная работа считается зачтенной после её устной защиты у преподавателя.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №1
по дисциплине «Параллельное программирование»

**Исследование особенностей реализации
классического алгоритма умножения матриц**

Выполнил: студент группы ФИБ-4301-51-00 _____ / **И.И. Иванов** /

Проверил: ст. преподаватель каф. ПМиИ _____ / В.А. Бызов /