ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ  
за курсом «Архітектура і програмне забезпечення»  
студент(а/ки) групи ПА-25-1  
Негря Дмитро Іванович

кафедра комп’ютерних технологій, ДНУ  
2025/2026

1. Постановка задачі (з Додатку А)  
   За допомогою мови C/C++ написати програму з наступним функціоналом:
   1. Функція створення вектора розміром N (універсального для вектор-стовпця і для вектор-рядка);
   2. Функція створення квадратної матриці розміром N на N;
   3. Функція множення вектора на квадратну матрицю;
   4. Функція множення квадратної матриці на вектор;
   5. Функції знищення вектора та квадратної матриці;

Зробити можливість створювати матриці з такими типами даних:

1. int
2. float
3. double

Для кожного типу відповідно зробити функції множення.

Після функцій для роботи з матрицями зробити тестування функціоналу:

1. Протестувати функціонал на малих розмірах (N=4, N=8, N=16), щоб переконатись у правильності роботи.
2. Протестувати функціонал на великих розмірах (N>1000), щоб дізнатись швидкість алгоритмів.
3. Експериментальним шляхом знайти максимум для N, поступово збільшуючи його на 1000, починаючи з N=1000 (наприклад, максимальний N=46000)

У вигляді додатків додати до звіту:

1. Додаток Б: Код всіх файлів проєкту;
2. Додаток В: Опис характеристик комп’ютера;
3. Додаток Г: Результати тестування функціоналу у вигляді таблиць з такими стовпцями:
4. N;
5. Створення квадратної матриці, с;
6. Створення вектора, с;
7. Перемноження квадратної матриці на вектор, с;
8. Перемноження вектора на квадратну матрицю, с;

Зробити 3 таких таблиці для кожного з типів: int, float, double;

Зробити графіки з порівняннями даних різних типів, та помістити поруч з таблицями в одному файлі.

1. Опис розв’язку + блок-схема

Проєкт містить декілька ключових елементів:

* 1. Папка bin – для виконуємого файла program.exe;
  2. Папка src – для вихідних файлів програми;
  3. Файл Makefile – для збірки проєкту:

Папка src містить такі вихідні файли:

1. *“mat\_int.c”* – фукнціонал для роботи з квадратною матрицею та вектором у типі даних int;
2. *“mat\_float.c”* – фукнціонал для роботи з квадратною матрицею та вектором у типі даних float;
3. *“mat\_double.c”* – фукнціонал для роботи з квадратною матрицею та вектором у типі даних double;
4. *“mat\_int.h”*, “*mat\_float.h”*, “*mat\_double.h”* – хедер файли для функціоналу;
5. *“mat.h”* – хедер файл з усіма типами матриць;
6. *“tests.c”* – файл з тестами;

У кожному з *“src/mat\_<type>.c”* файлі визначено однаковий функціонал, тільки для різних типів. Для прикладу візьмемо за основний тип float. Основні 2 функції, у яких виконується множення:

void mat\_mul\_1df\_2df(Mat1Df\* mat1d, Mat2Df\* mat2d, Mat1Df\* result\_mat);

void mat\_mul\_2df\_1df(Mat2Df\* mat2d, Mat1Df\* mat1d, Mat1Df\* result\_mat);

Інші функції (для створення, заповнення, друку та видалення) не мають у собі алгоритмів і виконують базові операції для підготовки та послідуючому прибиранні робочого місця з матрицями, тому окремо представлятися не будуть. А ось функції множення потребують блок-схеми, оскільки виконують прямі дії над матрицями:

|  |  |
| --- | --- |
| *Рис 1. Блок-схема множення квадратної матриці на вектор.* | Изображение выглядит как текст, диаграмма, зарисовка, круг  Содержимое, созданное искусственным интеллектом, может быть неверным.  *Рис 2. Блок-схема множення вектора на квадратну матрицю.* |

Блок-схеми виглядають майже однаково, оскільки формула для множення виглядає майже однаково, різниця тільки в зміні положень рядків і стовпців.

Для типів int та double функції аналогічні.

Хедер-файл *“src/mat.h”* містить в собі об’єднання всіх типів матриць за допомогою препроцесору \_Generic(). Це дозволяє передавати у такі функції, як: mat\_fill, mat\_print, mat\_mul, будь-які типи матриць(Mat1Di, Mat2Di, Mat1Df, Mat2Df, Mat1Dd, Mat2Dd) у будь-якій послідовності та автоматично отримувати потрібну функцію.

1. Вихідний текст програми розв’язку задачі (основні фрагменти з коментарями)

Див Додаток Б.

1. Опис інтерфейсу програми (керівництво користувача)

Інтерфейс цієї програми – функціонал для роботи з квадратною матрицею та універсальним вектором, а користувач може змінювати тільки спосіб тестування цього функціоналу напряму в файлі “*src/tests.c”,* змінюючи код.

1. Опис текстових прикладів

Матриці заповнюються автоматично числами від 0 до 10, проте це можна змінити за необхідності у “*src/tests.c”*.

Результат роботи програми: див. Додаток Г.

1. Аналіз помилок (опис усунення зауважень)

Під час написання програми помилок виявлено не було.

1. Висновки

У цій лабораторній роботі було розглянуто множення квадратної матриці на вектор та навпаки. Результати показали, що набагато швидше помножити квадратну матрицю на вектор ніж навпаки, через особливості кешування та оптимізації C та C++.

ДОДАТКИ

Додаток А.

У цьому додатку написане завдання лабораторної роботи.

**Лабораторна робота № 1**

Завдання: На мові С/C++ напишіть програму яка буде перемножувати квадратну матрицю розміром NxN на вектор розмірність N. Величину N виберіть таким чином, щоб максимально використати всю можливу оперативну пам’ять. Обчисліть затрати часу на дану дію. Перемножте також вектор на матрицю. Порівняйте затрачений час в обох випадках. Зробіть висновки. Запропонуйте зміну алгоритму таким чином, щоб затрачений час був приблизно однаковий.

Пояснення: Найшвидшим способом обходу є прямий послідовний. Це означає, що після звернення у програмі до деякого елемента відбувається звернення до елемента, що у пам'яті йде слідом за ним. Розглянемо розміщення у пам'яті двовимірного масиву у програмі мовою С/C++.

float A[N][N];

Відомо, що в мові С/C++ масиви розташовуються в пам'яті рядками (спочатку йдуть елементи першого рядка, потім елементи другого рядка і т.д.). Отже, у пам'яті він розміститься так:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **…** | A0,0 | A0,1 | A0,2 | … | A0,N-1 | A1,0 | A1,1 | A1,2 | … | A1,N-1 | … | AN-1,0 | AN-1,1 | AN-1,2 | … | AN-1,N-1 | **…** |

Отримуємо два варіанти перебору елементів масиву:

|  |  |
| --- | --- |
| **Швидко:**  **for** (i=0;i<N;i++)  **for** (j=0;j<N;j++) A[i][j]=x; | **Повільно:**  **for** (j=0;j<N;j++)  **for** (i=0;i<N;i++) A[i][j]=x; |
| **Швидко:**  **for** (i=0;i<N;i++)  **for** (j=0;j<N;j++) A[i][j]=x; | **Повільно:**  **for** (i=0;i<N;i++)  **for** (j=0;j<N;j++) A[j][i]=x; |

Розглянемо задачу перемноження двох квадратних матриць N×N. Якщо безпосередньо запрограмувати відому формулу: , наприклад, мовою С/C++, отримаємо наступний фрагмент програми:

for (i=0;i<N;i++)

for (k=0;k<N;k++)

for (j=0;j<N;j++) C[i][k]+=A[i][j]\*B[j][k];

Зауважимо, що у цьому випадку масив A перебирається рядками, а масив B – по стовпцям (дивимося на внутрішній цикл). Знаючи, що масиви в мові С/C++ зберігаються рядками, приходимо до висновку, що елементи масиву A перебираються послідовно, а елементи масиву B – ні. В даному випадку порядок обходу масиву C практично не важливий, оскільки між зверненнями до різних його елементів проходить багато часу. Щоб прискорити програму, потрібно, щоб принаймні у внутрішньому циклі елементи масивів перебиралися послідовно. Для цього необхідно або заздалегідь транспонувати масив B, або переставити цикли таким чином:

for (i=0;i<N;i++)

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

for (k=0;k<N;k++) C[i][k]+=A[i][j]\*B[j][k];

**За результатами виконання лабораторної роботи скласти звіт:**

**Виміряти час перемноження матриці на вектор, вектора на матрицю, а також час оптимізованого способу перемноження вектора на матрицю. Виміряти час для розмірів матриці від N=1000 до максимального розміру матриці, який вдалося розмістити в оперативній пам'яті з кроком 1000.**

* **До звіту додати скріни або опис характеристик обчислювальної техніки на якій виконувався чисельний експеримент**
* **Додати до звіту скріни про те, що перемноження відбувається вірно на прикладі невеликих матриць.**
* **Виконати перемноження та засікти час для різних типів даних: int, float, double.**
* **Створити таблиці, куди занести результати засічок часу.**
* **Створити графіки, на яких відображати дані з таблиць.**
* **Порівняти час перемноження різних типів даних та різних способів перемноження**

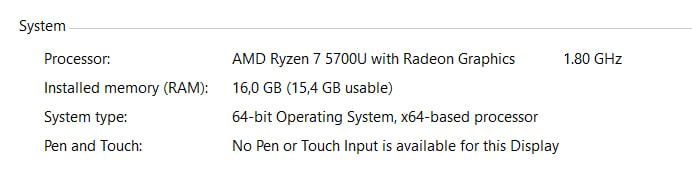
Додаток Б.

У цьому додатку написані всі файли з кодом проєкта.

Вихідний текст програми знаходиться в репозиторії GitHub: https://github.com/MikaCat2008/Architecture\_and\_software\_1

Додаток В.

У цьому додатку зображено опис характеристик комп’ютера.

  
*Рис 1. Опис характеристик комп’ютера.*

*  
Рис 2. Версія gcc.*

*Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рис 3. Версія make.*

Додаток Г.

У цьому додатку відображено результати роботи програми.

1. Тестування функціоналу для N=4, N=8, N=16 знаходяться у файлі *“результати.txt”*;
2. Таблиці та графіки знаходяться у файлі *“результати.xlsx”;*