<u>כפל מטריצות</u> - יש לנו 2 מטריצות ריבועיות באותו הגודל ורוצים לבצע כפל בינן.

ניתן לבצע את פעולה גם בקלות, ע"י 3 לולאות מקוננות. היתרון של הקוד הזה שהוא פשוט. החסרון שלו הוא בעיות.

שפר הראשון ניתן לבצע בלולאה הפנימית. ההשמה מתבצעת ל- C[i][j], שינוי תלוי בשתנים מהלולאה השלישית. לכן, אנחנו מבצעים הרבה חישובים מיותרים של המקום של C[i][j] בצורה.

כדי לסכם את הכל, יש לחשב משתנה כפל בלולאה השנייה (sum), פעם החישוב בלולאה השלישית ותבצע כתיבה. לאחר סיום הלולאה השלישית נבצע השמה של sum ל- C[i][j].

כדי לבדוק האם השפר באמת הטבה על זמן החישוב, נשתמש בשני שעונים ונבדוק חיסכון.

(*) כדי למנוע אילוץ בחירת דרך לחישוב עדיף פונקציה מעוימת ויתן להשים מס המוך moon.

דרך פשוטה יותר היא ע"י שימוש בכלי Profiling אנחנו נשתמש בכלי בסיסי ומוכנה בשם gprof.

בדרך תמיד נשים את הדגל g- . בעת שימוש ב-Profiling נוסיף את הדגל g- בדגל pg- .

כדי לבדוק, נוסף את הפקודה " less " בכפל בקובץ הרגיל.

<u>שימוש ב- cache עיקרי בידע</u>

הרעיון שעומד מאחורי ה- cache הוא שעשה תוכנה של גישה בהתבצעות הרבה פעמים ה- cache לזכרון נרצה נגישות יותר מהיר מה cache (שיפור מהירות).

החסרון של ה- cache הוא בכך שהכל בו המהירות הזו יחסית יקרים לכן, בדרך אין הרבה מקום ב- cache.

<u>כתיבת קוד ידידותי ל- cache</u>

אנו מניחים שמעמיס יש מקום ל- β מילים, בדיוק אי אפשר לסכם את כל הנתונים במטריצה. נניח כי גודל המטריצה N מתחלק ב- β יכן שהמעמיס לא יכל לרכוז שורה אמיתית שלמה.

אם נכתוב תוכנית שעוברת על המטריצה פעם אחת, לאחר כל קריאה יהיה לנו miss (כי הכתובת למעמיס הבא, פעם הראשון ובאחרון המטריצה נמצאת בכלוב שלא שייכת). לעומת זאת, אם נכתוב מטריצה שסורקת פעם שלישית, יהיה לנו miss רק לאחר כל β מילים.

u.multinet.co.il