**פרויקט גמר**

**למילוי חלקי של הדרישות לקבלת תואר הנדסאי**

**הנדסת תוכנה**

**בהתמחות: מחשבים**

**נושא הפרויקט: Business calculator**

**שם הסטודנטית: דסי שפירא**

**העבודה בוצעה בהנחיית: הגב' כהן**

תוכן

[מבוא 3](#_Toc104412000)

# מבוא

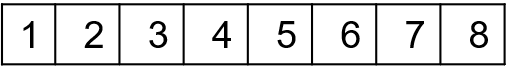
על מנת להבין ולהתעמק בנושא המהותי של חישוב מקבילי ומבוזר ניסיתי לחשוב על נושא שישלב את שתי הבחינות – חישוב מקבילי ומבוזר. ולכן שילבתי בפרויקט את הבחינה של מקביליות, ואפשרתי על ידי threads לכמה חישובים בו זמנית. ובנוסף את הבחינה של המבוזר, שרת לקוח.

פרויקט זה מחשב בדך יעילה הוצאות והכנסות של לקוח למשך שנה, בכל אחד מחדשי השנה, ע"י הזנת הנתונים לשנה הקרובה, וסכום השקעה ראשוני.

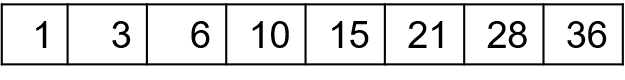
הפרויקט מתבסס על אלגוריתם prefix.

בהינתן סדרת מספרים, מערך ה- prefix sum שלה הוא מערך בו כל תא מכיל את סכום כל המספרים הנמצאים בסדרת המספרים לפניו, כולל הוא בעצמו.

לדוגמה:



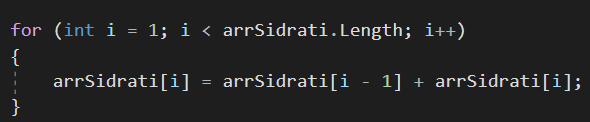
סדרת מספרים



Prefix sum

# תהליך המחקר

Prefix-sum

ראשית, מצאתי את האלגוריתם הסדרתי לחישוב ה- prefix sum. הוא פשוט ביותר:

כעת ניתן לחשוב, כיצד ניתן למקבל את האלגוריתם.

# מטרות

* לימוד והתנסות בכתיבת אלגוריתם ממוקבל ומבוזר.
* כתיבת תוכנית המבצעת באופן יעיל, חישוב prefix sum.

# תיאור האלגוריתם הנבחר

חישוב האלגוריתם הממוקבל יתבצע בשלושה מהלכים:

* כל ליבה / תהליכון מחשב את ה- prefix sum של האיברים במחיצה שלו באופן מקומי.
* האיברים האחרונים, שהם גם הגדולים יותר, של כל תהליכון- מועברים למערך עזר, כאשר המקום הראשון במערך מאותחל ל-0.
* כל תהליכון מעדכן את האיברים במחיצה שלו ע"י הוספת הערך המצוי במערך העזרת בהתאמה.

## תרשים זרימה של האלגוריתם:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 24 | 15 | 7 | 15 | 9 | 4 | 6 | 3 | 1 |

תהליכון 0

תהליכון 0

תהליכון 0

|  |  |  |
| --- | --- | --- |
| 24 | 15 | 6 |

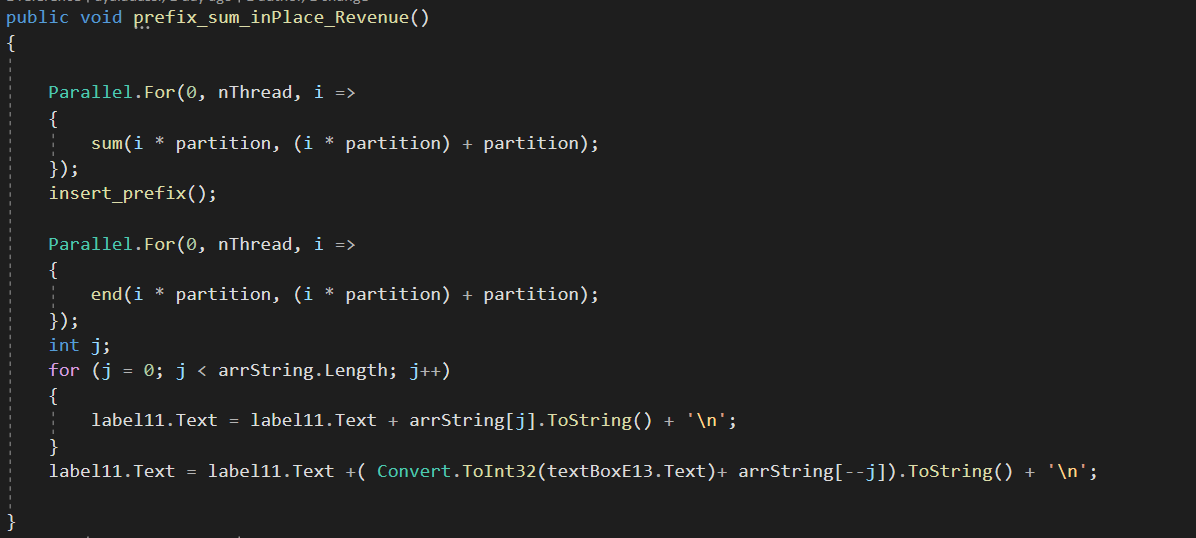
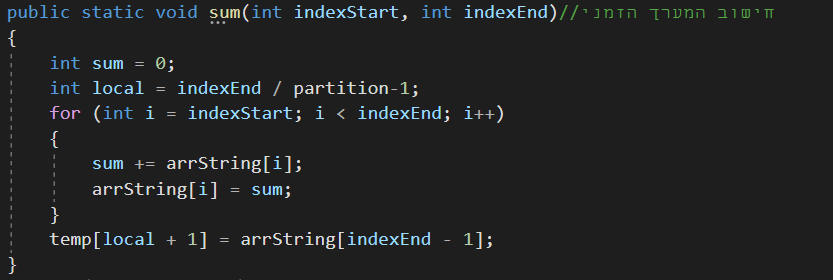
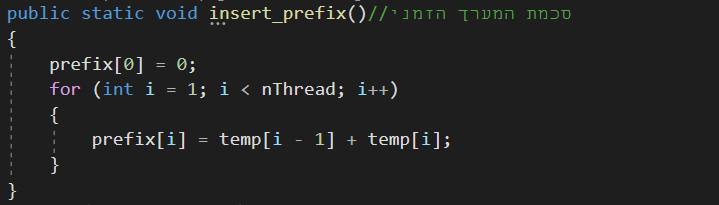
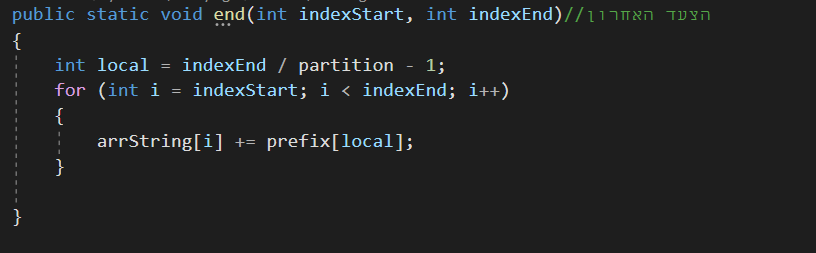
|  |  |  |
| --- | --- | --- |
| 21 | 6 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 45 | 36 | 28 | 21 | 15 | 10 | 6 | 3 | 1 |

# אפיון פונקציונלי

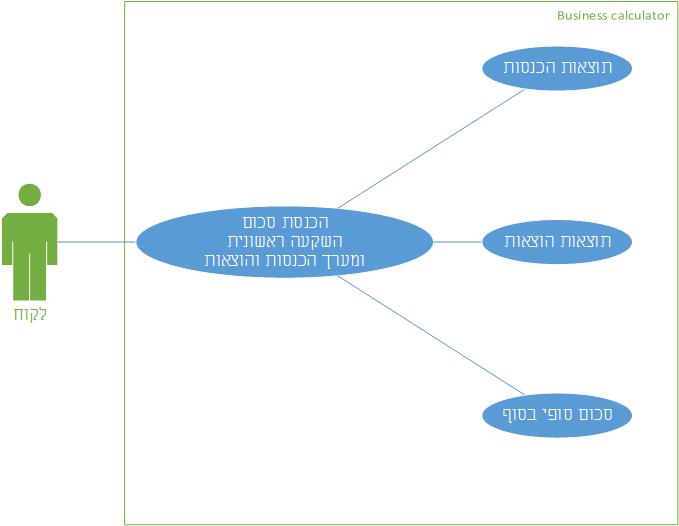
* Prefix\_sum\_inPlace\_Revenue()
* Sum\_Revenue()
* Insert\_prefix\_Revenue()
* End\_Revenue()
* Prefix\_sum\_inPlace\_Expenses()
* Sum\_Expenses()
* Insert\_prefix\_Expenses()
* End\_Expenses()

# קוד



ההוצאות באותו רעיון רק ממערך אחר.

# תרשיםMUL



# הוראות למשתמש

הלקוח מכניס סכום השקעה ראשוני, הכנסות והוצאות של בית ההעסק לשנה זו.

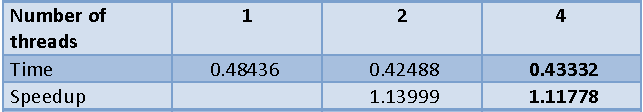
בחירה על "תוצאות הכנסות" תראה למשתמש מה יהיו ההכנסות שלו בכל אותה השנה.

בחירה על "תוצאות הוצאות" תראה למשתמש מה יהיו ההוצאות שלו בכל אותה השנה.

בחירה על "סכום סופי בסוף" תראה למשתמש מה יהיו הכנסות והוצאות שלו באותה השנה.

# מסכים

# תוצאות ההרצה:



השיפור שמושג עם שתי ליבות הוא רק 13%, ואילו עם ארבע ליבות- 11% בלבד, עקב התקורה הכרוכה במקביליות.

למעשה, במקרה שלפנינו רוב הזמן שנמדד הוא תקורה מקבילית, ורק חלקו הקטן הוא זמן חישוב נטו.

# מבני נתונים בהם משתמשים בפרויקט

מערך אחד המייצג את הסדרה החשבונית, ומערך נוסף- המכיל את ה- prefix sumses.

# מסקנות

עם מעט מחשבה, ניתן לייעל חישובים מסובכים- ולהופכם לקלים וקצרים הרבה יותר. גם אם המאמץ גדול יותר, התוצאה- שווה.

# פיתוחים עתידיים

אם נוסיף פונקציה המבצעת עבודת סרק ובכך מגדילה את זמן החישוב, יעילות המקבול תגדל כפי שנראה בטבלה הבאה:

