

# תרגול 4 הצפנה ופענוח

## רקע

בשיעור למדנו על עיקרון הירושה שעוזר לנו לחסוך קוד וליצור מבנה היררכי מסודר למחלקות התוכנית שלנו. דיברנו גם על הסתרת פונקציות, וראינו איך שכאשר יש למחלקה הירושה פונקציות עם אותו שם כמו במחלקת הבסיס היא מסתירה את הפונקציה של האב.

## מטרה

בתרגיל נעזור לשירות החשאי של מגשימים לבנות תוכנה לפיענוח צפנים 🕵️ נלמד כמה דרכים להצפין ולפענח הודעות.



אלה השלבים שנעבור:

הוספת תפריט	הוספת תמיכה באופרטור <<	עבודה עם קבצים	שימוש בשדות סטטיים	מימוש המחלקות	כתיבת שלד
הוספת תפריט עבור הפעולות בתוכנית.	הוספת תמיכה באופרטור	קריאה וכתיבה לקבצים	ספירת המופעים שנוצרו	ShiftText CaesarText Substitution	למידה עצמית UML יצירת מחלקות התרגיל

נתרגל מיומנויות חשובות:

- נתרגל ירושה בין מחלקות והסתרת פונקציות
- נתרגל עבודה עם קבצים
- נתרגל שימוש במשתנים סטטיים ופונקציות סטטיות
- מיומנות

את התרגיל צריך להגיש ב-GIT: [לינק להוראות שימוש ב-GIT](#).

כדאי לקרוא גם [דגשים לתכנות נכון](#).



"PRACTICE MAKES PERFECT"

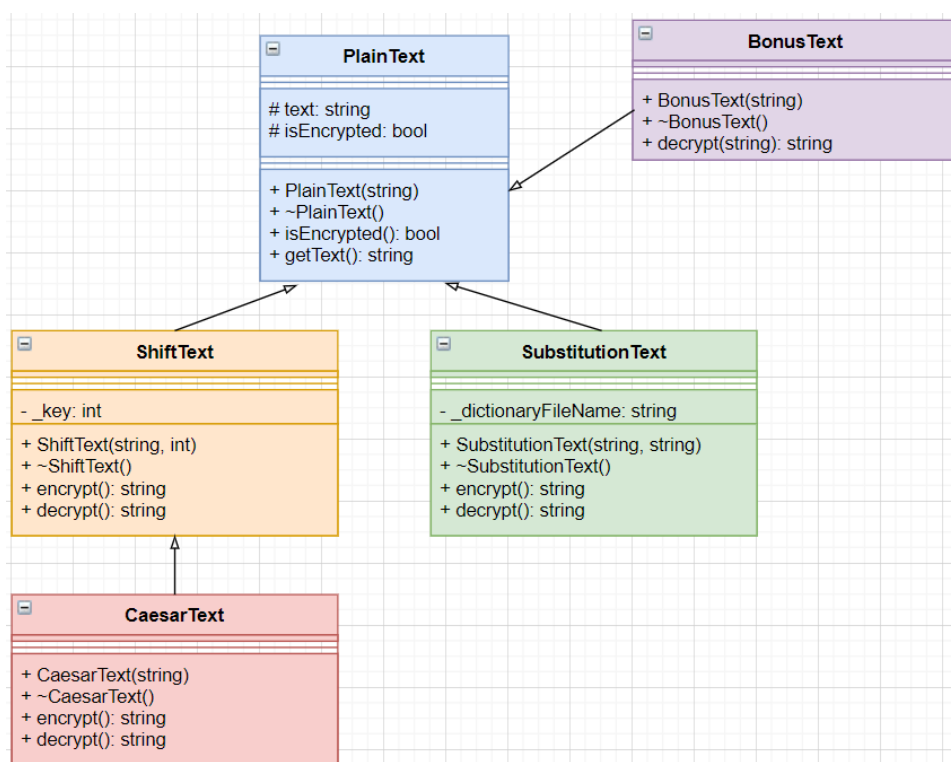
**בהצלחה יא אלופות ואלופים!**

## שלב 1: כתיבת מחלקות על פי תרשים UML

שלב 6	שלב 5	שלב 4	שלב 3	שלב 2	שלב 1
הוספת תפריט	הוספת תמיכה באופרטור <<	עבודה עם קבצים	שימוש בשדות סטטיים	מימוש המחלקות	כתיבת שלד

UML (Unified Modeling Language) היא שפה תקנית לעיצוב מונחה עצמים. אנשים רבים משתמשים ב-UML כדי להציג תרשימים של מערכות ותוכנות, וכיום השימוש ב-UML מאוד נפוץ גם ע"י מפתחי תוכנה, וגם ע"י בעלי תפקידים שעוזרים למדל ולתכנן מוצרי תוכנה.

אחד מהאלמנטים הנפוצים ב-UML הוא תרשים מחלקות (באנגלית Class Diagram). קראו על [תרשים מחלקות בוויקיפדיה](#) ובנו את קבצי ה-header של מחלקות התרגיל בהתאם לתרשים המחלקות הבא:



שימו ,

- תרשים מחלקות UML הוא כללי, ולא מספק מידע על מימוש בשפת תכנות מסוימת, לכן חשוב להוסיף בעצמנו את הסימונים והקונבנציות שאליהם אנחנו רגילים (לדוגמא const ו-reference איפה שצריך).

## שלב 2: מימוש המחלקות

שלב 6 הוספת תפריט	שלב 5 הוספת תמיכה באופרטור <<	שלב 4 עבודה עם קבצים	שלב 3 שימוש בשדות סטטיים	<b>שלב 2 מימוש המחלקות</b>	שלב 1 כתיבת שלד
----------------------	-------------------------------------	-------------------------	--------------------------------	--------------------------------	--------------------

קראו כיצד עובדות ההצפנות השונות [בקישור הבא](#) וממשו את הפעולות המפורטות בטבלאות הבאות:


### 1. PlainText

טקסט רגיל - PlainText	
מתודה/פעולה	תיאור
PlainText( <b>string</b> text)	יוצר מופע של הודעה בסיסית (בנאי)
~ PlainText()	ניקוי ושחרור זיכרון (מפרק)
<b>bool</b> isEncrypted()	מחזיר האם המחרוזת מוצפנת
<b>string</b> getText()	מחזיר את המחרוזת מהשדה _text


### 2. ShiftText

צופן היסט - ShiftText	
מתודה/פעולה	תיאור
ShiftText( <b>string</b> text, <b>int</b> key)	יוצר מופע של הודעה מוצפנת באמצעות הפונקציה encrypt ומפתח key (בנאי)
~ShiftText()	ניקוי ושחרור זיכרון (מפרק), במידה וצריך.
<b>string</b> encrypt( <b>string</b> text, <b>int</b> key)	פונקציה <b>סטטית</b> שמקבלת text ומפתח, מצפינה את הטקסט לפי האלגוריתם. הפונקציה מחזירה את הטקסט המוצפן.
<b>string</b> decrypt( <b>string</b> text, <b>int</b> key)	פונקציה <b>סטטית</b> שמקבלת text ומפתח, מפענחת את הטקסט לפי האלגוריתם. הפונקציה מחזירה את הטקסט המפוענח.
<b>string</b> encrypt()	מתודה שמצפינה את ה-text של המחלקה ומשנה את השדה isEnc בהתאם. המתודה מחזירה את הטקסט המוצפן.
<b>string</b> decrypt()	מתודה שמפענחת את ה-text של המחלקה ומשנה את השדה isEnc בהתאם. המתודה מחזירה את הטקסט המפוענח.

### CaesarText 3.

	Caesar Text - צופן היסט בעל מפתח קבוע
מתודה/פעולה	תיאור
CaesarText (string text)	יוצר מופע של הודעה מוצפנת באמצעות המתודה encrypt ומפתח קבוע עם הערך 3 (בנאי)
~CaesarText ()	ניקוי ושחרור זיכרון (מפרק), במידה וצריך.
string encrypt(string text)	פונקציה סטטית שמקבלת text, ומצפינה את הטקסט לפי האלגוריתם. הפונקציה מחזירה את הטקסט המוצפן.
string decrypt(string text)	פונקציה סטטית שמקבלת text, ומפענחת את הטקסט לפי האלגוריתם. הפונקציה מחזירה את הטקסט המפוענח.
string encrypt()	מתודה שמצפינה את ה-text של המחלקה ומשנה את השדה isEnc בהתאם. המתודה מחזירה את הטקסט המוצפן.
string decrypt()	מתודה שמפענחת את ה-text של המחלקה ומשנה את השדה isEnc בהתאם. המתודה מחזירה את הטקסט המפוענח.

### SubstitutionText 4.

	SubstitutionText – צופן החלפה
SubstitutionText(string text, string dictionaryFileName)	יוצר מופע של הודעה מוצפנת באמצעות הפונקציה encrypt ומפתח. (בנאי)  המפתח במקרה זה הוא מילון אותו קיבלתם בקובץ <b>dictionary.csv</b> הפונקציה מקבלת את שם הקובץ
~SubstitutionText()	ניקוי ושחרור זיכרון (מפרק)
string encrypt(string text, string dictionaryFileName)	פונקציה סטטית שמקבלת text ומפתח, ומצפינה את הטקסט לפי האלגוריתם. הפונקציה מחזירה את הטקסט המוצפן.
string decrypt(string text, string dictionaryFileName)	פונקציה סטטית שמקבלת text ומפתח, ומפענחת את הטקסט לפי האלגוריתם. הפונקציה מחזירה את הטקסט המפוענח.
string encrypt()	מתודה שמצפינה את ה-text של המחלקה ומשנה את השדה isEnc בהתאם. המתודה מחזירה את הטקסט המוצפן.
string decrypt()	מתודה שמפענחת את ה-text של המחלקה ומשנה את השדה isEnc בהתאם. המתודה מחזירה את הטקסט המפוענח.

בסוף התרגיל ישנו נספח עם דוגמאות, אם תרצו תוכלו לראות דוגמא לכל אלגוריתם (הצפנה ופענוח)

## הנחיות והערות חשובות:

- בהוראות לא רשומות החתימות המדויקות, אבל האחריות שלכם/ן היא להוסיף `const` ולקבל פרמטרים כ-reference היכן שצריך.
- ההצפנות יקבלו אותיות קטנות באנגלית בלבד
- ההצפנות ידלגו על רווח, פסיק ונקודה (ישאירו אותם כמו שהם).
- יש בסה"כ 26 אותיות ב-'abc' האנגלי
- יש לעבוד עם תווי `ascii` ואין צורך לעבוד עם `unicode`.
- בקובץ מסוג CSV הערכים מופרדים בפסיק (Comma Seperated Value)
- יש לבדוק את תקינות המחלקות ע"י הצפנה ופיענוח של טקסטים שנמציא בעצמנו, או לוודא שהכל תואם לדוגמאות שנמצאות בסוף המסמך.
- באלגוריתמים שבהם יש גם הצפנה וגם פענוח צריך להתקיים  $Decrypt(Encrypt(message)) = message$
- במתודות `encrypt`, `decrypt` של המחלקה `SubstitutionText` יש לפתוח קובץ (עם השם שסופק למחלקה). מומלץ להשתמש [במחלקה `fstream`](#) כדי לקרוא ולכתוב לקבצים.
- חשוב לבדוק שניתן לקרוא לפונקציות ההצפנה/פענוח הסטטיות גם מבלי ליצור מופע של המחלקה.
- מומלץ לבדוק את תקינות המחלקות ע"י הרצת הקובץ `conversation.cpp`

## שלב 3: שימוש בשדות סטטיים

שלב 6 הוספת תפריט	שלב 5 הוספת תמיכה באופרטור <<	שלב 4 עבודה עם קבצים	שלב 3 שימוש בשדות סטטיים	שלב 2 מימוש המחלקות	שלב 1 כתיבת שלד
----------------------	-------------------------------------	-------------------------	--------------------------------	------------------------	--------------------

עברו על [המצגת שנמצאת בתיקיית התרגיל](#) העוסקת במשתנים סטטיים והוסיפו את השדה `numOfTexts`.

`numOfTexts` הוא שדה סטטי שמטרתו לספור כמה מופעים של המחלקה `PlainText` (או היורשים ממנה) נוצרו לאורך התכנית.

חשבו באיזה מתודה נגדיל את המשתנה כדי שנצטרך לכתוב את שורת הקוד פעם אחת בלבד ולא לספור דברים פעמיים.

הגדירו את המשתנה כ-`public`, כך שניתן יהיה לגשת אליו ישירות בלי מתודת `get`.

דוגמא שבסופה ערך המשתנה יהיה 5:

```
int main()
{
    PlainText p1("blabla"); // count is 1
    ShiftText s1("Yalla yalla", 4); // count is 2
    PlainText p2("zoom zoom zoom"); // count is 3
    CaesarText c1("abcdefghi"); // count is 4
    SubstitutionText sb1("boo foo zoo", "dictionary.csv"); // count is 5
}
```

## שלב 4: עבודה עם קבצים

שלב 6 הוספת תפריט	שלב 5 הוספת תמיכה באופרטור << >>	שלב 4 עבודה עם קבצים	שלב 3 שימוש בשדות סטטיים	שלב 2 מימוש המחלקות	שלב 1 כתיבת שלד
----------------------	--	-------------------------	--------------------------------	------------------------	--------------------

בחלק הזה ניצור מחלקה עם 3 מתודות סטטיות שקשורות לקבצים.

FileHelper	
תיאור	מתודה/פעולה
<b>מתודה סטטית:</b> פותחת את הקובץ באמצעות השם שניתן, ומחזירה את תכולת הקובץ כ- <code>std::string</code>	<code>string readFileToString(string fileName)</code>
<b>מתודה סטטית:</b> המתודה מקבלת שני ארגומנטים: - שם קובץ input (קלט) - שם קובץ Output (פלט) הפונקציה תקרא מהקובץ input מילה אחר מילה ותכתוב את המילים לקובץ output, כל מילה בשורה נפרדת	<code>void writeWordsToFile(string inputFileName, string outputFileName)</code>
<b>מתודה סטטית:</b> המתודה מקבלת שני ארגומנטים: - טקסט (מחרוזת) - שם קובץ Output (פלט) הפונקציה תרשום את ה-text לקובץ output.	<code>void saveTextInFile(string text, string outputFileName)</code>

### הנחיות והערות חשובות:

- כתבו main.cpp שבו תבדקו את תקינות הפונקציות על הקובץ example.txt.
- אם נוצר קובץ, וודאו שיש לו את ההרשאות הנכונות (כתיבה/קריאה), ושהוא נוצר בצורה תקינה.
- בפונקציה writeWordsToFile, כדי לקרוא מהקובץ או לכתוב לקובץ יש להשתמש באופרטורים << >> של Output Stream **ואין להשתמש** בפונקציה getline
- זכרו לבדוק שקובץ נפתח כמו שצריך, ולסגור את הקובץ בסיום.





## שלב 5: האופרטור <<

שלב 6 הוספת תפריט	שלב 5 הוספת תמיכה באופרטור <<	שלב 4 עבודה עם קבצים	שלב 3 שימוש בשדות סטטיים	שלב 2 מימוש המחלקות	שלב 1 כתיבת שלד
----------------------	-------------------------------------	-------------------------	--------------------------------	------------------------	--------------------

בחלק זה נוסיף את האופרטור '<<' למחלקות ההצפנה.  
האופרטור מאפשר (בין השאר) הדפסה של אובייקטים בעזרת `std::cout`.

ממשו את האופרטור ככה שכאשר האובייקטים של המחלקות יישלחו ל-`std::cout` יודפסו שם האלגוריתם והטקסט שבמחלקה **אחרי הצפנה**.

לדוגמא:

בעת הרצת הקוד הבא:

```
int main()
{
    CaesarText c1("abcdefghi");
    std::cout << c1 << std::endl;
}
```

יודפס למסך:

```
Cesar
defghijklm
```

## הנחיות והערות חשובות:

- כדי להצהיר על האופרטור מומלץ לקרוא על מתודות מסוג "friend"
- במידה למחלקה אין פונקציית פענוח (כמו PlainText למשל), יש להדפיס את הטקסט כמו שהוא.
- מומלץ להיעזר בפתרון של תרגיל מס' 3 במידה ותרצו לראות איך מממשים את האופרטור (המשימה הזו ניתנה כבונוס בתרגיל הקודם).

## שלב 6: תפריט

שלב 1	שלב 2	שלב 3	שלב 4	שלב 5	שלב 6
בתיבת שלד	מימוש המחלקות	שימוש בשדות סטטיים	עבודה עם קבצים	הוספת תמיכה באופרטור <<	הוספת תפריט

בחלק האחרון של התרגיל נבנה תפריט המאפשרת למשתמש לבצע 4 פעולות:

### 1. הצפנה ופענוח מחרוזת

- יש לקלוט מהמשתמש מחרוזת שברצונו להצפין או לפענח, ואלגוריתם שבו התוכנית תשתמש (Shift, Caesar, Substitution)
- יש להציג למשתמש את התוצאה של המחרוזת המוצפנת או המפוענחת במסך

### 2. הצפנה ופענוח קובץ טקסט

- יש לקלוט מהמשתמש שם של קובץ טקסט (.txt).
- המשתמש יוכל לבחור אם לפענח או להצפין את תכולת הקובץ ע"פ האלגוריתמים שמימשנו:
  - צופן הזזה עם מס' שהמשתמש מספק
  - צופן קיסר
  - צופן החלפה
- יש להציג למשתמש שתי אופציות:
  - שמירת קובץ חדש שבו יהיה הטקסט המפוענח/המוצפן
  - הצגת הטקסט על המסך

### 3. הדפסת מספר הטקסטים שהתקבלו בתכנית

- יש להשתמש בשדה הסטטי שיצרנו במשימה 3

### 4. יציאה

## הנחיות והערות חשובות:

- כאשר אובייקט נוצר הוא מיד מוצפן, חשבו איך ניתן לפענח הודעה באמצעות אלגוריתם מסוים מבלי ליצור מופע של המחלקה. (רמז: סטטי)
- אחרי פיענוח של קובץ/מחרוזת התפריט יחזור למסך הראשי שם המשתמש יוכל לפענח קובץ/מחרוזת נוספת.
- ניתן להניח שהגודל המקסימלי של המחרוזת הנקלטת הוא **1024** תווים.
- הקפידו לבדוק את הקלט של המשתמש
  - זכרו שאנחנו עובדים עם אותיות קטנות בלבד ותווים מיוחדים (פסיק, רווח, נקודה) שעליהם אנחנו מדלגים בפיענוח/הצפנה.
- נסו לשמור על התפריט פשוט ככל הניתן

## שלב בונוס: אתגרים נוספים

שלב 6	שלב 5	שלב 4	שלב 3	שלב 2	שלב 1
הוספת תפריט	הוספת תמיכה באופרטור <<	עבודה עם קבצים	שימוש בשדות סטטיים	מימוש המחלקות	כתיבת שלד

### שלב בונוס אתגרים נוספים



השבוע משימת הבונוס מורכבת משני חלקים לא קשורים:

#### 1. אתגר פענוח (קשה)

כתבו מחלקה חדשה בשם BonusText אשר מבצעת פיענוח בלבד והשתמשו בה כדי לפענח את תכולת הקובץ "encrypted.txt" בשביל לפענח את הקובץ תצטרכו להיות יצירתיים...

#### רמז:

הקובץ מכיל אותיות באנגלית, ומוצפן באמצעות צופן החלפה (כל אות מוחלפת באות אחרת). חישבו איך אפשר לנצל את זה.

**!** אנא הגישו את הקוד שבו השתמשתם כדי לפענח את הקובץ.

## 2. הפניית ערוץ

אחד הדברים שאפשר לעשות עם stream-ים הוא להפנות stream אחד ל stream אחר. הוסיפו קובץ חדש בשם Bonus.cpp ובו יש פונקציה בשם redirect\_cout.

- הפונקציה פותחת קובץ חדש בשם "log.txt", במידה והקובץ קיים הפונקציה צריכה לפתוח קובץ חדש במקומו (לדרוס).
- הפונקציה תבצע הפניה של cout ככה שבמקום להדפיס למסך, הדברים אשר יישלחו לcout באמצעות האופרטור << יכתבו לקובץ log.txt.

**רמז:**

קיראו על rdbuf, כדאי לראות דוגמאות קוד

**!** אנא הגישו main נפרד שמכיל שימוש בפונקציה שכתבתם/ן.

## נספחים

### דוגמאות לשימוש במחלקות הצופן

#### CaesarText (הסטה של 3 אותיות)

```
int main()
{
    CaesarText csr("roses are red, my name is dave, this makes no sense, microwave.");
    std::cout << csr.getText();
}
```

#### ShiftText (הסטה של 7 אותיות)


```
int main()
{
    ShiftText shft("an apple a day keeps anyone away if you throw it hard enough.", 7);
    std::cout << shft.getText();
}
```

## SubstitutionText (עם המילון שסופק בקבצי התרגיל)

```
int main()
{
    SubstitutionText sub("sometimes when i close my eyes, i cant see.", "dictionary.csv");
    std::cout << sub.getText();
}
```

## הגשה ב-GIT

- את הפרויקט יש לנהל ב-Git, לפתוח repository חדש בתוך קבוצת ה-gitlab שלנו ושל המדריך/ה, ולהגיש לינק לפרויקט ב-NEO (אפשר לעשות comment עם הלינק או להגיש מסמך txt עם הלינק בפנים).
- יש להעלות ל-repository את כל הקבצים הרלבנטיים לתרגיל (קבצי txt, מסמכים, ומשאבים אחרים שבהם השתמשנו). חשוב להעלות את פרויקט ה-Visual Studio השלם ולהתעלם מקבצים לא נחוצים ([הנחיות במסמך הבא](#)), במידה ולא הועלה הפרויקט השלם, אין להעלות את שאר הקבצים שיוצר Visual Studio – הם רבים מאוד, הם לא מכילים מידע נחוץ להרצת הפרויקט אצל המדריך, ורק יוצרים בלגן.
- הבחירה אילו קבצים להעלות ל-repository נעשית באמצעות הפקודות add ו-rm. אופציה נוספת (מומלצת) היא להוסיף קובץ gitignore. אשר יתעלם מהקבצים הלא נחוצים. במידה ותרצו תוכלו להיעזר ב[סרטוני עזר בנושא GIT](#).
- כסיימתם/ן, בדקו שניתן להריץ את הפרויקט בקלות – בצעו Clone אל תיקייה במחשב אשר שונה מזו שעבדתם/ן, ותראו שהפרויקט נפתח ע"י לחיצה על קובץ ה-sln ויכול לרוץ בלי בעיה

 Ex4.sln

## דגשים:

- את הפרויקט יש לפתוח בקבוצת ה-gitlab שאליה משותף/ת המדריך/ה כ-Maintainer.
- יש לוודא שכל הקבצים הרלבנטיים נוספו ל-repository (באמצעות הפקודה add), במידת הצורך ניתן להוריד קבצים מיותרים (באמצעות הפקודה rm)
- יש לבצע commit עבור כל סעיף, ובנקודות שבהן הוספנו שינויים חשובים (לפי הדגשים שהועברו בכיתה).
- עבור כל commit, זכרו לכתוב הודעה קצרה ואינפורמטיבית, שאפשר יהיה להבין מה היה השינוי בקוד.

- יש לדחוף את הקוד (באמצעות הפקודה push) ל-repository בסיום העבודה שלנו, חשוב שבסיום העבודה שלנו, ובמידה ונפנה למדריך/ה, ב-repository יהיה הקוד המעודכן ביותר.
- במידה ושכחנו או שאנחנו לא בטוחים איך מעלים קובץ, או מתעלמים מקבצים, כדאי לצפות בסרטוני ההדרכה בנושאי GIT. ניתן לגשת לסרטונים בלשונית ה-resources שבכיתת ה-NEO.
- בסיום העבודה יש להגיש לכיתת ה-NEO קישור ל-repository.

## כללי

1. יש לבדוק שכל **המטלות מתקמפלות ורצות ב-VS2022**. מטלה שלא תעבור קומפילציה אצל הבודק לא תיבדק **והניקוד שלה יהיה 0** 😞
2. יש לבדוק שהקוד שכתבתם עובד. יש להריץ בדיקות שלכם ולוודא שהקוד ברמה טובה.
3. כאשר אתם מתבקשים לממש פונקציה, ממשו בדיוק את הנדרש. אין להוסיף הדפסות וכדו'. אם הוספתם תוך כדי הבדיקות שלכם הדפסות, אנא דאגו להוריד אותם לפני ההגשה.
4. להזכירכם! העבודה היא עצמית, ואין לעשות אותה ביחד.
5. על כל שאלה או בעיה יש לפנות למדריך, לפחות 36 שעות לפני מועד ההגשה.

## דגשים לתכנות נכון

- כדאי לקמפל כל מספר שורות קוד ולא לחכות לסוף! הרבה יותר קל לתקן כאשר אין הרבה שגיאות קומפילציה. בנוסף קל יותר להבין מאיפה השגיאות נובעות.
- כדאי לכתוב פונקציה ולבדוק אותה לפני שאתם ממשיכים לפונקציה הבאה. כלומר, כתבו תכנית ראשית שמשתמשת בפונקציה ובודקת האם היא עובדת כראוי. חישבו על מקרי קצה ונסו לראות מה קורה.
- בכל פעם שאתם מתקנים משהו, זכרו שיכול להיות שפגעתם במשהו אחר. לכן עליכם לבדוק שוב מהתחלה.
- חשפו החוצה רק את הממשק המינימלי הדרוש (minimal API), הגדירו את שדות המחלקה כפרטיים, וכמה שפחות מתודות כציבוריות.