

# תרגול 10 סל מוצרים

## רקע

בתרגיל הקודם ובתרגילים שעשינו לאורך הסמסטר מימשנו מבני נתונים מסוגים שונים כדי להבין לעומק איך הם יכולים לעזור לנו ביצירת סדר ושיפור ביצועים בתוכניות שלנו. אז עכשיו שאנחנו מבינים איך הדברים עובדים, אין באמת סיבה שנממש בעצמנו מבני נתונים לתוכניות שנכתוב בהמשך ובדיוק בגלל זה למדנו על ספריית STL (Standard Template Library) 📖 📦

הספרייה מכילה מימושים מוכנים של מבני נתונים, אלגוריתמים וגם הגדרות נוספות שמאוד יכולות לעזור לנו בשיפור הקוד שלנו. המחלקות והפונקציות של הספרייה נבדקו ונמדדו ע"י מתכנתים מוכשרים ועכשיו כל מה שנשאר לנו זה פשוט להשתמש בה 😊

## מטרה

בעזרת המבנים של **STL** נממש מערכת שתשמש את חנות מגשימים העומדת להיפתח בקרוב 🛒🔒. המערכת תהייה מורכבת מחנויות, רשימת לקוחות, סל לכל לקוח ורשימת מוצרים לכל סל. לשם כך נזדקק לכמה מחלקות בסיסיות, למבנים `std::map`, `std::set`, `std::deque` ובמידת הצורך לכמה אלגוריתמים מהספרייה **Algorithm**.



אלה השלבים שנעבור:

מדידת ביצועים שלב בונס	מימוש תפריט	מימוש מחלקת Customer	מימוש מחלקת Store	מימוש מחלקת Item
<ul style="list-style-type: none"> <li>השוואה בין Containers שונים</li> </ul>	<ul style="list-style-type: none"> <li>ניהול קלט ופלט</li> </ul>	<ul style="list-style-type: none"> <li>שימוש ב-std::map</li> <li>שימוש ב-std::set</li> <li>שימוש ב-iterators</li> </ul>	<ul style="list-style-type: none"> <li>שימוש ב-std::deque</li> <li>שימוש ב-comparators</li> <li>שימוש באלגוריתמים של ספריית STL</li> </ul>	<ul style="list-style-type: none"> <li>העמסת אופרטורים</li> <li>שימוש ב-enums</li> </ul>

נתרגל מיומנויות חשובות:

- שימוש במבני נתונים מספריית STL
- שימוש באלגוריתמים של STL
- עבודה עם איטרטורים
- בניית ממשק נוח למשתמש (תפריט)
- 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: מוצרים

מימוש מחלקת Item	מימוש מחלקת Store	מימוש מחלקת Customer	מימוש התפריט main.cpp	שלב בונס השוואת ביצועים
---------------------	----------------------	-------------------------	--------------------------	----------------------------

### מחלקת Item (מוצר בחנות)

בשלב הראשון נממש מחלקה המייצגת מוצר/פריט בחנות שלנו.

לרשותנו הקובץ **Item.h** שבו מוגדר **enum** בשם **ItemCategory**, ופונקציה סטטית **getItemCategoryString(const ItemCategory type)** אשר מחזירה מחרוזת שמייצגת כל קטגוריה (לדוגמא עבור קטגוריית HOME הפונקציה תחזיר "Home") והיא נועדה לעזור לנו בהדפסות מוצרים בהמשך.


תחילה נגדיר את השדות הבאים


Item	
שדה	תיאור
<code>string _name</code>	שם המוצר
<code>string _serialNumber</code>	מס' סידורי
<code>int _count</code>	מס' הפריטים בסל המוצרים מהסוג הזה
<code>double _unitPrice</code>	מחיר יחידה
<code>ItemCategory _category</code>	קטגוריית המוצר (ע"פ ה-enum שניתן בקובץ)



שימו 

- השדה `_count` מגדיר **כמה פריטים** מהסוג הזה נמצאים בסל המוצרים של הלקוח.
- מחיר המוצר הוא עבור יחידה **אחת**.
- המס' הסידורי משמש כמפתח **השוואה** בין מוצרים, ועליו להיות 5 תווים בלבד.

 Item	
מתודה/פעולה	תיאור
<pre>Item(string name, string serialNumber, double unitPrice, ItemCategory category)</pre>	<p>בנאי</p> <p>שימו 🧡, במידה וקרו אחד מהמקרים הבאים:</p> <ul style="list-style-type: none"> <li>• והמס' הסידורי של הפריט הוא לא 5 תווים</li> <li>• המחיר של הפריט שלילי</li> </ul> <p>הבנאי זורק <code>std::invalid_argument</code> והאובייקט לא יבנה</p>
<pre>double totalPrice()</pre>	<p>מחזירה את המחיר הכולל של הפריטים השמורים</p> <p><math>(unitPrice * count)</math></p>
<ul style="list-style-type: none"> <li>• <code>string getName()</code></li> <li>• <code>string getSerial()</code></li> <li>• <code>double getPrice()</code></li> <li>• <code>int getCount()</code></li> <li>• <code>ItemCategory getCategory()</code></li> </ul>	<p>Getters</p>
<pre>void setCount(int newCount)</pre>	<p>משנה את הערך של השדה <code>_count</code></p> <p>שימו 🧡, במידה וניסו לשנות את השדה <code>count</code> לערך שלילי יש לזרוק <code>std::invalid_argument</code></p>
<pre>bool operator &lt;(const Item&amp; other) const</pre>	<p>אופרטור &lt;</p> <p>מחזיר true אם <b>המספר הסידורי</b> של המוצר קטן מהמספר הסידורי של המוצר <code>other</code></p>
<pre>bool operator &gt;(const Item&amp; other) const</pre>	<p>אופרטור &gt;</p> <p>מחזיר true אם <b>המספר הסידורי</b> של המוצר גדול מהמספר הסידורי של המוצר <code>other</code></p>

 Item (המשך...)	
מתודה/פעולה	תיאור
<pre>bool operator ==(const Item&amp; other) const</pre>	אופרטור == מחזיר true אם המספר הסידורי של המוצר שווה למספר הסידורי של המוצר other
<pre>friend std::ostream&amp; operator&lt;&lt;(std::ostream&amp; os, const Item&amp; item);</pre>	אופרטור << מאפשר הדפסה של האובייקט ל-output stream בפורמט הבא: [Serial: <serial>, Name: <name>, Category: <category>, Price: <price>, Amount: <count>] ניתן לראות דוגמת הדפסה <a href="#">בחלק של הנספחים</a>

אחרי הגדרת השדות נעבור להגדרת ומימוש המתודות:

שימו 

- ניתן להוסיף מתודות נוספות, אך חובה להגדירן ב-`private`
- בהוראות לא רשומות החתימות המדויקות, האחריות שלכם/ היא להוסיף `const` ולקבל פרמטרים ב-reference היכן שצריך.
- אסור לשנות את החתימות של המתודות בקובץ ה-header (מלבד הוספת מתודות פרטיות)
- בתיקיה של קבצי התרגיל תוכלו למצוא קובץ בדיקות בשם `Test1Item.cpp`, שבו תוכלו להשתמש כדי לבדוק את הגדרות ומימוש המחלקה שלכם/.



## שלב 2: חנויות

שלב בונס השוואת ביצועים	מימוש התפריט main.cpp	מימוש מחלקת Customer	מימוש מחלקת Store	מימוש מחלקת Item
----------------------------	--------------------------	-------------------------	----------------------	---------------------

### המחלקה Store

כעת נממש מחלקה אשר מייצגת **חנויות**.

לכל חנות יש רשימה של פריטים אשר מאוחסנים במבנה נתונים סדרתי של `std::deque` – STL. גם הפעם, הקובץ **Store.h** ניתן לנו מראש, ומכיל הגדרות שבהן ניתן להשתמש. בנוסף, ישנו גם הקובץ **Store.cpp** שבא עם מימוש מוכן של אחת המתודות של המחלקה.

1. בראש הקובץ תוכלו למצוא הגדרות ל-3 struct-ים. המבנים האלו יישמשו אותנו בהמשך כאשר נרצה למיין את רשימת המוצרים לפי קריטריונים שונים. Struct כזה נקרא בשפה מקצועית **Comparator**

קראו על comparators

[/https://www.geeksforgeeks.org/comparator-class-in-c-with-examples](https://www.geeksforgeeks.org/comparator-class-in-c-with-examples)



והשלימו את שורה 24 בתוך ה-struct `NameComparator`, היכן שרשום `/* Complete by yourself */` ככה שההשוואה בין הפריטים תתבצע על פי השם שלהם.

2. בתחתית הקובץ תוכלו לראות מתודת עזר פרטית `void getInventoryFromFile(const std::string inventoryFileName);`

המתודה קוראת את המידע שמוגדר בקבצי ה-csv שניתנו יחד עם התרגיל, ומכניסה כל שורה כפריט (`Item`) אל תוך ה-`deque` של המחלקה. המתודה **כבר מומשה עבורנו** בקובץ **Store.cpp** דוגמאות לשימוש:

```
getInventoryFromFile("InventoryPharm.csv"); ○  
getInventoryFromFile("InventoryIKEA.csv"); ○  
getInventoryFromFile("InventorySuperMarket.csv"); ○
```

יש להגדיר את המתודות הבאות במחלקת `Store`:

<div>  Store </div>	
מתודה/פעולה	תיאור
<pre>Store(std::string storeName,       std::string inventoryFileName);</pre>	<p>בנאי</p> <p>שימו  , בתוך הבנאי יש לקרוא למתודה <code>getInventoryFromFile</code></p>
<pre>std::string getName()</pre>	<p>מחזירה את שם החנות</p>
<pre>std::string getSortedProductList(     SortingCriteria sortingCriteria)</pre>	<p>המתודה מקבלת אחת מקטגוריות המיון המוגדרות בראש הקובץ, ומחזירה מחרוזת של רשימת פריטים ממיונת ע"פ הקריטריון.</p> <p>הקריטריונים האפשריים:</p> <p><code>CATEGORY, NAME, PRICE, SERIAL</code></p>
<pre>std::string getProductListFilteredByCategory(     ItemCategory category);</pre>	<p>המתודה מקבלת אחת מקטגוריות הפריטים המוגדרות בקובץ <code>Item.h</code>, ומחזירה מחרוזת של רשימת פריטים השייכים לקריטריון הזה בלבד.</p> <p>הקריטריונים האפשריים:</p> <p><code>FOOD, PHARM, CLEANING, HOME</code></p>
<pre>friend std::ostream&amp; operator&lt;&lt;(std::ostream&amp; os, const Item&amp; item);</pre>	<p>אופרטור &lt;&lt;</p> <p>מאפשר הדפסה של האובייקט ל-output stream בפורמט הבא:</p> <p>Store name: &lt;name&gt;</p> <p>Products:</p> <p>[0] - &lt;item0&gt;</p> <p>[1] - &lt;item1&gt;</p> <p>...</p> <p>[n] - &lt;itemn&gt;</p> <p>ניתן לראות דוגמת הדפסה <a href="#">בחלק של הנספחים</a></p>

<pre>Item operator[] (const int itemNumber) const;</pre>	<p>אופרטור גישה [ ]</p> <p>מאפשר קבלת פריט (Item) מתוך רשימת הפריטים של המחלקה ע"פ אינדקס.</p> <p>במידה והוכנס אינדקס שלא בגבולות ה-deque יש לזרוק <code>std::invalid_argument</code></p> <p>דוגמה:</p> <pre>Store s1("Shefa Isaschar",         "InventorySuperMarket.csv"); Item a = s1[9];</pre>
--	--

## הנחיות והערות חשובות:

- בהוראות לא רשומות החתימות המדויקות, האחריות שלכם/ן היא להוסיף `const` ולקבל פרמטרים כ-reference היכן שצריך.
- אסור לשנות את החתימות של המתודות בקובץ ה-header (מלבד הוספת מתודות פרטיות)
- זיכרו לעשות `<algorithm>` `#include` כדי שתוכלו להשתמש באלגוריתמים של STL.
- את המתודה הממיינת והמסננת, יש לממש באמצעות אלגוריתמים של ספריית STL.
  - את המיונים יש לבצע באמצעות `std::sort` ([קריאה נוספת כאן](#)).
  - השתמשו ב-comparator-ים שהוגדרו בראש הקובץ.
  - את הסינון מותר לבצע איך שתמצאו, אבל מומלץ להתנסות בפונקציות `std::copy`, `std::copy_if` שיחסכו לכם/ן המון שורות קוד. מאוד מומלץ [לקרוא על תחביר lambda](#) ולשלב אותו בקוד שלכם
  - כדי לאתגר את עצמכם/ן, נסו לעשות את הסעיפים ללא לולאות בכלל 🙌





## שלב 3: לקוחות וסלי מוצרים

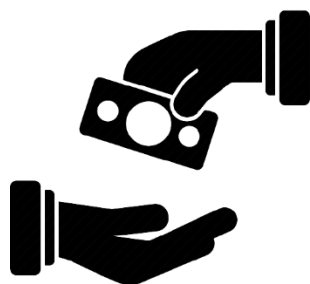
שלב בונוס השוואת ביצועים	מימוש התפריט main.cpp	מימוש מחלקת Customer	מימוש מחלקת Store	מימוש מחלקת Item
-----------------------------	--------------------------	-------------------------	----------------------	---------------------

### המחלקה Customer



נעבור להגדרת מחלקה המייצגת **לקוח** של החנויות שלנו.  
הלקוח יכול להחזיק כמה סלי מוצרים מחנויות שונות.

תחילה נגדיר את השדות הבאים:

Customer	
שדה	תיאור
<code>string _name</code>	שם הלקוח
<code>map&lt;string, set&lt;Item&gt;*&gt; _items</code>	מילון שמחזיק את סלי המוצרים של הלקוח. סל מוצרים הוא מבנה נתונים <code>std::set&lt;Item&gt;</code> סלי המוצרים של הלקוח מוחזקים בתוך <code>std::map</code> בכל רשומה ב- <code>map</code> , המפתח שבעזרתו ניתן לגשת לכל סל מוצרים זה שם הסל (מסוג <code>std::string</code> ) והערך שחוזר הוא <b>מצביע</b> לסל המוצרים.



הוסיפו למחלקה את המתודות הבאות:


	Customer
מתודה/פעולה	תיאור
Customer( <b>string</b> name)	בנאי
Customer()	בנאי ברירת מחדל מגדיר לקוח עם ערכי default (שם ריק)
~Customer()	מפרק
<b>void</b> createNewShoppingCart( <b>string</b> shoppingCartName);	יוצרת סל חדש עם השם הנתון. שימו  , נדרשת הקצאת זיכרון דינמית ( <b>new</b> ). אחרי פעולת המתודה צריך להיווצר סל מוצרים חדש ריק שאליו ניתן לגשת דרך ה- <b>std::map</b>
<b>string</b> getName()	מחזירה את שם הלקוח
<b>double</b> shoppingCartSum( <b>string</b> shoppingCartName) <b>const</b> ;	מחזירה את החשבון (סכום המחירים) של הפריטים בסל המוצרים ששמו הועבר למתודה
<b>double</b> totalSum()	מחזירה את החשבון הכולל של כל הפריטים <b>בכל</b> סלי המוצרים יחד
<b>set&lt;Item&gt;*</b> getShoppingCart( <b>string</b> shoppingCartName) ;	מחזירה את סל המוצרים עם השם שהועבר למתודה



## Customer (המשך...)

תיאור	מתודה/פעולה
מוסיפה פריט לסל המוצרים של הלקוח. במידה והפריט קיים, המתודה תעלה את ערכו של השדה <code>_count</code> של הפריט ב-1.	<pre>void addItem(Item item, string shoppingCartName)</pre>
מסירה פריט מסל המוצרים של הלקוח. במידה ויש כמה פריטים מאותו הסוג, המתודה תוריד את ערכו של השדה <code>_count</code> של הפריט ב-1.	<pre>void removeItem(Item item, string shoppingCartName)</pre>

### הנחיות והערות חשובות:

- בהוראות לא רשומות החתימות המדויקות, האחריות שלכם/ היא להוסיף `const` ולקבל פרמטרים כ-`reference` היכן שצריך.
- אסור לשנות את החתימות של המתודות בקובץ ה-`header` (מלבד הוספת מתודות פרטיות)
- שימו , מבני הנתונים `std::set` ו-`std::map` לא מאפשרים כפילויות (אין אפשרות להכניס שני פריטים עם אותו מפתח)
- המתודה `shoppingCartSum` חייבת להיות מוגדרת `const`
- כשניגשים ל-`map` מתוך מתודה `const` לא ניתן להשתמש באופרטור `[]`. יש להשתמש במתודה `at` במקום, שאליה מעבירים את המפתח.  
לקריאה נוספת: <https://en.cppreference.com/w/cpp/container/map/at>

## שלב 4: תפריט וממשק משתמש

שלב בונוס השוואת ביצועים	מימוש התפריט main.cpp	מימוש מחלקת Customer	מימוש מחלקת Store	מימוש מחלקת Item
-----------------------------	--------------------------	-------------------------	----------------------	---------------------

### סעיף א' – כתיבת תפריט

בתיקיית התרגיל תוכלו למצוא את הקובץ `main.cpp` שבו מוגדר `std::map` בשם `abcCustomers`. **תזכורת:** `std::map` הוא כמו מילון שמחזיק מיפוי בין מפתח לערך. במקרה שלנו המפתח הוא **שם הלקוח**, הערך הוא האובייקט `Customer` המחזיק את סלי המוצרים שלו. בנוסף, ב-`main` תוכלו לראות יצירה של 3 חנויות שונות מתוך קבצי ה-`csv` שניתנו יחד עם התרגיל. כיתבו תפריט שבעזרתו משתמש יוכל להיכנס למערכת כדי לנהל סלי מוצרים ולהוסיף/להסיר מוצרים מ-3 חנויות שונות. למשתמש יוצג תחילה המסך הבא:

Welcome to MagshiMart!

1. To sign as customer and buy items
2. To update existing customer's items
3. To print the customer who pays the most
4. To exit

### אופציה 1 – הרשמת משתמש חדש

במידה ומשתמש בחר באופציה הראשונה נקלוט את שם המשתמש שלו. לאחר מכן נציג למשתמש אופציה ליצירת סל מוצרים חדש.

1. To create a new shopping cart
2. Back to main menu

במידה והמשתמש בחר ליצור סל חדש, נבקש את שם הסל וניצור סל חדש

Enter the new shopping cart name:

במידה והשם קיים נציג שגיאה ונחזור לתפריט הראשי, במידה ולא, נבקש מהמשתמש לבחור אחת מתוך 3 החנויות.

Select the store to buy from:

1. to buy in Shefa Isaschar (Super Market)
2. to buy in MagshiKEA (Home products)
3. to buy in MagshiPharm (Pharm and utils)
4. Back to main menu

אחרי שהמשתמש בחר חנות, נציג את רשימת המוצרים והמספרים שלהם, ונקלוט ממנו את המוצר שברצונו לקנות עד אשר יקיש 0 (בלולאה).

**דוגמא** עם שני מוצרים:

The items you can buy are: (0 to exit)

1. Milk 3% price: 5.17
2. Olive Oil price: 29.9

What item would you like to buy? Input:

שימו 

- במידה ומשתמש הקליד שם שכבר קיים יש להדפיס הודעת שגיאה ולהחזירו לתפריט הראשי.
- במידה ומשתמש הוסיף כמה מוצרים מאותו הסוג יש לעדכן את שדה ה-count של המוצר בהתאם.
- אם הובנס קלט לא תקין יש להדפיס הודעת שגיאה ולהחזיר את המשתמש לתפריט הראשי.

**אופציה 2 – עדכון סל מוצרים של משתמש קיים**

במידה ומשתמש בחר באופציה השנייה נקלוט את שם המשתמש שלו.

אם המשתמש **לא קיים** נציג שגיאה ונחזור לתפריט הראשי

במידה והמשתמש קיים נציג את האופציות הבאות:

1. To create a new shopping cart
2. Update existing shopping cart
2. Back to main menu

אם המשתמש בחר **ליצור סל חדש** נקלוט את שם הסל ואז נבקש מהמשתמש לבחור אחת מתוך 3 החנויות (בדומה לחלק הקודם כאשר לקוח חדש נרשם פעם ראשונה ויצר סל חדש).

אם המשתמש בחר **לעדכן סל מוצרים קיים**, נציג למשתמש את סלי המוצרים שלו

Select Shopping Cart: (0 to exit)

1. NewHomeProducts
2. Groceries

בשני המקרים (גם אם נוצר סל חדש או מעדכנים סל קיים), נציג למשתמש שתי אופציות נוספות.

1. Add items
2. Back to main menu

במידה והמשתמש בחר **להוסיף פריטים** לסל המוצרים (אופציה 1) נציג למשתמש את המוצרים שהוא יכול להוסיף לסל (כמו בסעיף הקודם).

שימו 

- במידה ומשתמש הקליד שם שלא קיים יש להדפיס הודעת שגיאה ולהחזירו לתפריט הראשי.
- במידה ומשתמש הוסיף כמה מוצרים מאותו הסוג יש לעדכן את שדה ה-count של המוצר בהתאם.
- אם הוכנס קלט לא תקין יש להדפיס הודעת שגיאה ולהחזיר את המשתמש לתפריט שבו הוא היה (התפריט הראשי או תפריט העדכון של סל המוצרים)
- **בנוסף** אם תרצו לאתגר את עצמכם, הוסיפו למשתמש גם אופציה **להסיר פריטים** מסל המוצרים, וזו משימה קצת יותר מאתגרת כי תצטרכו לדאוג להציג את המוצרים שכבר קיימים בסל המוצרים של הלקוח, ולאפשר למשתמש למחוק אותם.

**אופציה 3 –** הדפסת הלקוח שמשלם הכי הרבה

הדפסו את שם הלקוח עם סל המוצרים היקר ביותר, וגם את המחיר שהוא שילם (totalSum).

## שלב בונוס: Benchmarking (מידת ביצועיים)

שלב בונוס השוואת ביצועים	מימוש התפריט main.cpp	מימוש מחלקת Customer	מימוש מחלקת Store	מימוש מחלקת Item
-----------------------------	--------------------------	-------------------------	----------------------	---------------------



### מידת ביצועים

לפני שמתחילים פרויקט גדול, ולפני שמתחילים לכתוב תוכנה חדשה, לפעמים נצטרך לחקור קצת ולהבין באילו משאבים הכי כדאי לנו להשתמש.

במשימת הבונוס של השבוע תתנסו במשימה שכמעט כל מתכנת/ת מתנסה בה – **השוואת ביצועים**. בבונוס נבנה מחלקות שיעזרו לנו להשוות בין מבני נתונים בספריית STL, ולראות מה היתרון והחסרון של כל **Container**.

הבונוס של השבוע לא פשוט, אבל מאוד מומלץ 🤖

### משימה 1 – בניית API משותף

צרו מחלקה אבסטרקטית טהורה בשם **Facade\***. מחלקה זו תהיה מחלקה אבסטרקטית ((שיש בה pure virtual functions) מטומפלטת, שמחלקות שנבנה בהמשך יירשו ממנה. המטרה היא שבתוך המחלקה נגדיר 2 מתודות שכל מי שיירש מהמחלקה יממש, והן יישמשו לצורך בדיקת הביצועים.

\*מחלקות מהסוג הזה נקראות **Facade**, מושג שמגיע מעולם האדריכלות ומתאר את הקידמה של בניין (החלק הקדמי שרואים). המושג **Facade** במדעי המחשב מתאר מצב שבו אנו יוצרים ממשק אחיד, ומצמצמים את המתודות רק לאלו שאנו צריכים, כל השאר לא מעניין אותנו. ([לקריאה נוספת על Facade pattern](#))

כל מחלקה שתירש את מחלקת **Facade** תצטרך לממש 2 פונקציות עיקריות:

**הוספה** - פונקציה שמקבלת איבר מסוג **T** (טמפלייט), מוסיפה אותו למבנה הנתונים, ומחזירה **true** אם האיבר נוסף בהצלחה.  
`bool add(const T item)`

**חיפוש** - פונקציה שמקבלת איבר מסוג **T** (טמפלייט), ומחזירה **true** אם הוא נמצא במבנה הנתונים.  
`bool contains(const T& item) const`

## משימה 2 – בניית מחלקות מבני נתונים

יש ליצור מחלקה עבור כל אחד ממבני הנתונים הבאים:  
מחלקת **SimpleLinkedList** - תחזיק רשימה מקושרת (`std::linked_list`)  
מחלקת **SimpleHashMap** - תחזיק טבלת גיבוב (`std::unordered_set`)  
מילון - **SimpleTreeSet** - תחזיק טבלה ממוינת (`std::set`)  
ווקטור - **SimpleVector** - תחזיק מערך דינמי (`std::vector`)  
תור דו כיווני - **SimpleDeque** - תחזיק תור דו כיווני (`std::deque`)

כל מחלקה צריכה להחזיק מופע שלו ולממש את הפונקציות `add`, `contains` באמצעות הפונקציות שחושף ה `container` מספריית STL.  
לדוגמא, במחלקה `SimpleLinkedList` יהיה שדה פרטי מהסוג `std::linked_list`, ושתי מתודות: המתודה `add` שבתוכה נקרא למתודה `add` של `std::linked_list`, והפונקציה `contains` שבתוכה נקרא למתודה `contains` של `std::linked_list`.

## משימה 3 – ביצוע בדיקות

בעזרת מחלקת ה **Facade** שמימשתם/ן, יש לבדוק ביצועים (כדאי במילי שניות) של מבני הנתונים.  
את המדידה יש לעשות באופן זהה עבור כל **אחד** מהמבנים, ולא לכלול דברים נוספים שעשויים להשפיע על התוצאות.  
יש לבצע את הבדיקות הבאות:

- מדידת **הוספת איבר** בכל מבנה נתונים:
  - הוספת כל האלמנטים (באמצעות המתודה `add`) מהקובץ **data1.txt** (מצורף) אחד אחרי השני לכל אחד מהמבנים.
  - הוספת כל האלמנטים (באמצעות המתודה `add`) מהקובץ **data2.txt** (מצורף) אחד אחרי השני לכל אחד מהמבנים.
  - יש למדוד כמה זמן לקח לכל אחד ממבני הנתונים, ולאחר מכן לסמן איזה מבנה נתונים היה הכי יעיל עבור קובץ **data1** ואיזה עבור **data2**.
  - מומלץ להשתמש בספריית **<chrono>**, במשתנים מסוג `clock_t` ובפונקציה `clock()`.
- מדידת **חיפוש איבר** בכל מבנה נתונים:
  - עבור המבנים שאליהם הוכנסו הנתונים מהקובץ `data1`, יש למדוד כמה זמן לקח לכל אחד למצוא את הערכים:
    - המחרוזת "hi"
    - המחרוזת "13170890158"
  - עבור המבנים שאליהם הוכנסו הנתונים מהקובץ `data2`, יש למדוד כמה זמן לקח לכל אחד למצוא את הערכים:
    - המחרוזת "hi"
    - המחרוזת "23"

למה לדעתכם/ן התוצאות שונות? למה במקרה הראשון מבנה נתונים מסוים יעיל יותר, ובמקרה השני הביצועים שלו גרועים יותר? 🤔

# בהצלחה!



## נספחים

### דוגמאות הדפסה ופלט מצופה

#### דוגמה לפלט מצופה מהדפסת Item


[Serial: SSStIo, Name: Baby Food Powder 900g, Category: Food, Price: 106.900000, Amount: 1]

#### דוגמה לפלט מצופה מהדפסת Store

```
Store name: IKEA
Products:
[0] - [Serial: SSStIo, Name: Baby Food Powder 900g, Category: Food, Price: 106.900000, Amount: 1]
[1] - [Serial: DTCx7, Name: Baby Wipes 24u, Category: Pharm, Price: 27.900000, Amount: 1]
[2] - [Serial: Qwxq7, Name: Diapers 40u, Category: Pharm, Price: 51.900000, Amount: 1]
[3] - [Serial: XLBCM, Name: Body Lotion 750ml, Category: Pharm, Price: 12.900000, Amount: 1]
[4] - [Serial: G0s5A, Name: Toilet Paper 24u, Category: Pharm, Price: 34.900000, Amount: 1]
[5] - [Serial: 3vIVS, Name: Toothpaste , Category: Pharm, Price: 15.900000, Amount: 1]
[6] - [Serial: boPnI, Name: Bleach 2 liter, Category: Cleaning, Price: 12.500000, Amount: 1]
[7] - [Serial: 10qgM, Name: Washing Machine powder 1.25Kg, Category: Cleaning, Price: 12.900000, Amount: 1]
[8] - [Serial: Jn2XA, Name: Washing Machine Conditioner 1 liter, Category: Cleaning, Price: 16.900000, Amount: 1]
[9] - [Serial: G9vKR, Name: Shampoo 750ml, Category: Pharm, Price: 10.900000, Amount: 1]
[10] - [Serial: 7f9b0, Name: Hair Conditioner 700ml, Category: Pharm, Price: 11.900000, Amount: 1]
[11] - [Serial: Fz8sA, Name: Deodorant 150ml, Category: Pharm, Price: 13.900000, Amount: 1]
[12] - [Serial: y3hMm, Name: Eye Contacts liquid 350ml, Category: Pharm, Price: 49.900000, Amount: 1]
[13] - [Serial: 8sZ4N, Name: Mouth Wash 500ml, Category: Pharm, Price: 36.900000, Amount: 1]
[14] - [Serial: RarYT, Name: Dishwash Soap 515ml, Category: Cleaning, Price: 13.900000, Amount: 1]
[15] - [Serial: SdQ46, Name: Window Cleaning Fluid 1 liter, Category: Cleaning, Price: 11.900000, Amount: 1]
[16] - [Serial: EVrno, Name: Paper Tows 6pack, Category: Cleaning, Price: 10.900000, Amount: 1]
[17] - [Serial: dVQyW, Name: Nail Polish 15ml, Category: Pharm, Price: 49.900000, Amount: 1]
[18] - [Serial: u10o0, Name: Man Perfume 100ml, Category: Pharm, Price: 250.000000, Amount: 1]
[19] - [Serial: g7WhS, Name: Woman Perfume 100ml, Category: Pharm, Price: 285.000000, Amount: 1]
```

## 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[בסרטוני עזר בנושא GIT](#).
- כסיימתם/ן, בדקו שניתן להריצ 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

 Ex10.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 שבכיתה ה-NEO.
- בסיום העבודה יש להגיש לכיתה 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.