סמינר בתכנות מערכות דפנסיבי

מגיש: אלון שנקלר 206280547

# תוכן הסמינר - בניית מערכת מאובטחת לביזור משימות המורכבת מרכיבים בשפות שונות

# חלק ראשון – תוכנת לקוח בשפת C++

## שפת C++ תכונות ומרכיבים

מקורות:

* Stroustrup, Bjarne. *The C++ programming language*. Pearson Education India, 2000.
* Lippman, Stanley B. *C++ Primer*. Pearson Education India, 2005.
* Smith, Ben. "Object-oriented programming." *AdvancED ActionScript 3.0: Design Patterns*. Apress, 2011. 1-25.

C++ היא שפת תכנות *לא ספציפית* – כלומר התוכנות אשר השפה מיועדת לשמש לכתיבתן אינן למטרה ספציפית, אלא כללית, לכל סוג תוכנה קיים.

עם זאת, היישום הנפוץ של השפה הוא בתחום *תכנות המערכות* [התחום העוסק בכתיבה של תוכנות אשר או דורשות ביצועים גבוהים (למשל משחקי מחשב או מערכות מוטמעות בחומרה) או תוכנות אשר אינן נגישות למשתמש ישירות ומשמשות כמצע\פלטפורמה ליישומי משתמש].

שפת C++ היא super-set של שפת C. כלומר, שפת C++ היא שפת C בתוספת יכולות ותחביר (עד כדי פרטים זניחים).  
אחת התופעות של תכונה זו היא שכל תכנית בשפת C, למעט מקרי קצה ייחודים, היא גם תכנית תקינה בשפת C++.

### תכנות פרוצדורלי

פרדיגמת התכנות הבסיסית ביותר הנתמכת ע"י שפת C++ היא *תכנות פרוצדורלי*.

Decide which procedures you want; use the best algorithms you can find. - Bjarne Stroustrup

פרדיגמת התכנות הפרוצדורלי מתארת את הפעולה של כתיבת תוכנה כרצף הבא:

1. מציאת האלגוריתמים המתאימים לפתרון בעיה נתונה
2. תרגום כל אלגוריתם בסיסי לפונקציה בשפת התכנות
3. כתיבת התוכנה כרצף של קריאות לפונקציות והעברת פרמטרים ותוצאות מקריאה לקריאה

כדי לבצע את הפעולה של העברת פרמטרים ותוצאות ממקום למקום אנו מגדירים *משתנים*, שהם צימוד בין שם בקוד לבין פיסת מידע.  
ב-C++ יש לציין את הטיפוס של כל פיסת מידע לצורך שימוש בה, וקיימת התאמה קרובה בין סוגי המידע (*הטיפוסים* השונים) הנתמכים לחומרה אשר התוכנה רצה עליה.

תכונה זו, יחד עם תכונות אחרות משייכות את C++ לקבוצת *Low-Level Programming Languages.*

לצורך תיאור אלגוריתמים בקוד ניתן לבצע על המשתנים השונים פעולות אריתמטיות ופעולות "מבחן" או *control flow* אשר מאפשרות לשנות את מסלול ריצת הקוד באמצעות בחינת ערך האמת של פסוקים אריתמטיים שונים.

שפות רבות מתנהלות בצורה דומה, ולכן נדלג על יתר הקונספטים הבסיסיים המשותפים לרוב שפות התכנות ונסתכל על תכונות ייחודיות יותר של C++.

### User-Defined Types

שפת C++ מאפשרת למתכנת להוסיף סוגי מידע חדשים שהוא יוכל לעבוד איתם בהמשך, באמצעות הגדרת טיפוסים חדשים עם המילה השמורה *class*.

המתכנת יכול לייצר טיפוס חדש באמצעות הרכבה של מספר טיפוסים בסיסיים ואוסף פעולות שהטיפוס החדש יתמוך בהן.

הטיפוסים שהמתכנת מגדיר בעצמו מתנהגים כמעט לחלוטין כמו הטיפוסים המובנים של השפה ומגדילים את ארגז הכלים הזמין למתכנת.

באמצעות המילה השמורה *virtual* ניתן להגדיר טיפוסים מופשטים, או ממשקים אשר מאפשרים להסתיר את פרטי המימוש של טיפוסים, ולחשוף למשתמש בהם רק את אוסף הפעולות שהן מאפשרות לו לבצע. תכונה זו מסייעת בהפיכת אותם טיפוסים חדשים לכלים יסודיים ולא לקוד שהמשתמש מכיר.

### תכנות מונחה עצמים

חיסרון בולט של אותם טיפוסים שמוגדרים ע"י משתמש שהצגנו בסעיף הקודם הוא חוסר הגמישות שלהם.  
לאחר הגדרת טיפוס שכזה, לא ניתן להוסיף לו יכולות ולכן אם יש לנו מספר טיפוסים בעלי תכונות והתנהגויות משותפות עלינו לשכפל את החלקים המשותפים בהגדרה שלהם.  
ב-C++ פותרים את הבעיה הזו באמצעות תמיכה ב*תכנות מונחה עצמים*.

תכנות מונחה עצמים הוא פרדיגמת תכנות אשר נתמכת ע"י C++.  
פרדיגמה זו מנחה את המתכנת להסתכל על הבעיה שהוא מנסה לפתור במונחים של העצמים שהבעיה מערבת והתכונות שלהם.  
אובייקט (בעברית *עצם*) הוא ישות בעלת אוסף פעולות שניתן להפעיל עליה, *מצב\תצורה* אשר משתנים בעקבות הפעלת פעולות וטיפוס.

ניתן לראות כי אותם User-Defined Types מתאימים להגדרת אובייקט!  
אבל, אותו חיסרון שתיארנו קודם נפתר בפרדיגמה של תכנות מונחה עצמים ע"י הרחבת הגדרת הטיפוס ל*מחלקה (class)* והגדרת קשרים בין מחלקות.  
קשרים אלו נקראים ירושה, ומאפשרים הרחבה של טיפוס קיים באמצעות הוספת טיפוסים בסיסיים למבנה המרוכב והוספת פעולות ש*תת-הטיפוס* יתמוך בהן, מעבר לפעולות המוגדרות ב*מחלקת האב*.

### Generic Programming

להשלים!