Mini-Project in File Management Systems

דו"ח פרויקט ומדריך למשתמש

מיני פרויקט בארגון וניהול קבצים

יאיר סטשבסקי 201643798 ymsil719@gmail.com
דניאל מילר 316212075 danielmiller20@gmail.com

תוכן

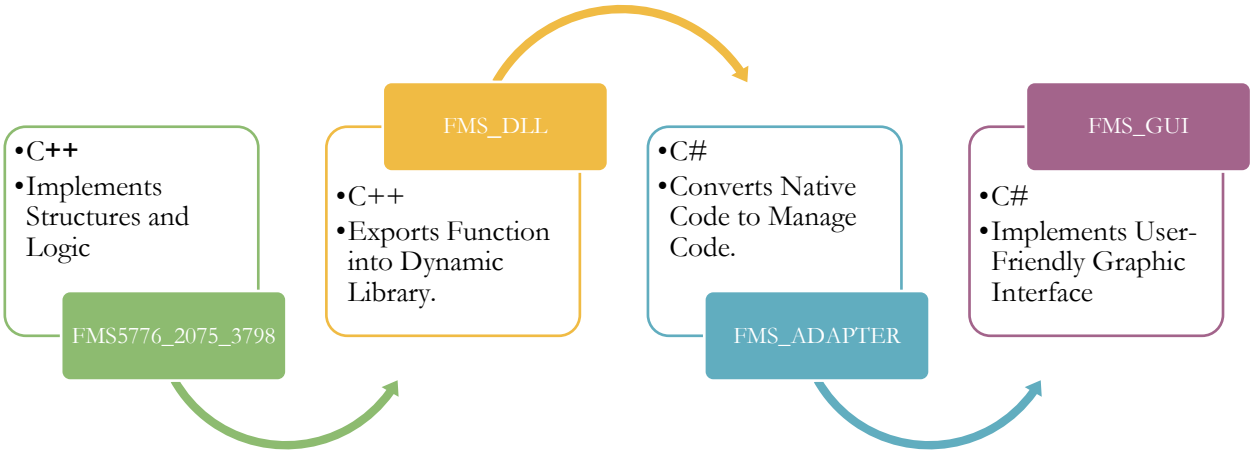
מבוא

מטרת הפרויקט היא לבנות מערכת המדמה את פעולת הדיסק הקשיח. תהליך המימוש כולל 5 שלבים(0-4) המפורטים בהמשך, וכולל גם ממשק ידידותי למשתמש אשר יקל על הגורם האנושי לבדוק ולהבין את המערכת.
מערכת הדיסק שבנינו היא בשפת ++C ואילו הממשק הגראפי שבחרנו עובד בWPF ולכן, המרנו את תשתית הדיסק לספריה סטאטית (.lib) כך שהיא תהווה תשתית לפרויקט כולו אשר יתממש בWPF ב-#C.
בנוסף, נבנה 2 פרויקטים נוספים אשר יעזרו ויקלו על ההתממשקות.

- פרויקט ראשון בשם "FMS_DLL" אשר נכתב בשפת ++C ויכיל פונקציה מקבילה לכל פונקציה שנרצה לחשוף מהתשתית. כל פונקציה תכיל הגדרות נוספות כגון כיצד להמיר משתנים מאותו סוג ומסוגים שונים בין השפות. פרויקט זה יוגדר כספריה דינאמית (.dll)
- פרויקט שני בשם "FMS_adapter" אשר יבנה בשפת #C ו"יתרגם"/יתאים את הפונקציות מ/אל "FMS_DLL". פרויקט זה יוגדר כclass library.

מבט כללי

כך ייראה מבנה הפרויקט באופן כללי:



תיאור השלבים בקצרה

ככלל, שלבים 0-3 התבצעו בפרויקט FMS5776_2075_3798 שהוא מממש את הדיסק המדומה עצמו. שלב 4 (FMS_DLL ו-FMS_Adapter) אחראי על מימוש מנגנון ההתממשקות בין קוד ה-++C לקוד ה-#C. בשלב 5 (או חלק ב' של שלב 4) בנינו את הממשק הגרפי (FMS_GUI).

שלב 0: הגדרה ויישום של מחלקות התשתית (מבנה + פעולות) המיישמות את הדיסק המדומה ברמה הפיזית שלו, כרצף של סקטורים: שדות ופונקציות של מחלקת Disk.

שלב 1: הגדרה ויישום של פונקציות תשתית נוספות, במסגרת המחלקה Disk.

שלב 2: יישום התשתית לטיפול בקבצים, מבחינה מבנית, פיזית, ברמת הדיסק: יישום יצירת והרחבת קובץ תוך הקצעת שטח דיסק בשבילו, ומחיקת קובץ תוך החזרת השטח שהוא תפס, לשטח הפנוי של הדיסק. (בשלב הזה מסתכלים על קובץ כרצף של סקטורים, ללא התייחסות לרמה הלוגית שבה מסתכלים על קובץ כרצף של רשומות).

שלב 3: להגדיר וליישם מחלקה בשם FCB (File Control Block) שתאפשר לטפל בקובץ מבחינת התוכן שבו, מבחינה לוגית, כרצף של רשומות: לנהל את כל פעולות הקלט/פלט הלוגיות (לפי רשומות) הקשורות לפתיחה כלשהי של קובץ שבדיסק.

שלב 4-5: מעבר ל-C# ובניית הממשק הגרפי בWPF.

שלב 0

שלב זה עוסק ברובד הפיזי של הדיסק. יצירת הדיסק, עיגון הדיסק, ניתוק הדיסק, קריאה וכתיבה לדיסק. לצורך כך הוגדרה המחלקה **Disk**, אשר בנויה ממבנים ומחלקות אשר הדיסק מכיל, **Sector**, **Cluster**, **DAT**, **Volume Header** וכד' המדמים את מבנה הדיסק האמיתי. נעשה שימוש במבנים אלו ע"מ ליצור הדמיה של דיסק אמיתי. תוך שימוש בפונקציות המדמות את השימושים הפיזיים בדיסק.

VHD – Volume Header

מבנה מתאר את כל המידע המבני של הדיסק.

יש לשים לב כי ה-VHD תמיד ישכון בסקטור הראשון של הדיסק ללא תלות במאפייני יצירתו.

תיאור שדות המבנה:

- **sectorNr** - מספר סידורי של הסקטור בדיסק.
- **diskName** - שם זיהוי הדיסק.
- **diskOwner** - שם בעל הדיסק.
- **prodDate** - תאריך יצור הדיסק.
- **ClusQty** - סה"כ יחידות הקצאה (clusters) בדיסק.
- **dataClusQty** - מספר יחידות הקצאה לנתונים בלבד.
- **addrDAT** - כתובת הסקטור שמכיל את ה- DAT.
- **addrRootDir** - כתובת ה- cluster שמכיל את התיקייה הראשית (Root Directory).
- **addrDATcpy** - כתובת הסקטור שמכיל עותק שני של ה-DAT.
- **addrRootDirCpy** - כתובת ה- cluster שמכיל עותק שני של התיקייה הראשית (Root Directory).
- **addrDataStart** - כתובת ה- cluster הראשון בדיסק המיועד לנתונים.
- **addrUserSec** - כתובת של הסקטור בו שמורים פרטי המשתמשים.
- **formatDate** - תאריך פרמוט.
- **isFormated** - האם כבר מפורמט? (TRUE/FALSE).
- **emptyArea** - שמור לשימוש עתידי- לא בשימוש כרגע.

תיאור פונקציות המחלקה:

VHD ()
בנאי ברירת מחדל שמאתחל את המחלקה VHD בערכי ברירת מחדל.

VHD (...)
בנאי שמאתחל את המחלקה VHD בערכים ספציפיים.

DAT – Data Allocation Table

מבנה זו משמשת לתיאור ה-cluster-ים התפוסים בדיסק כאשר הסיבית ה-i ערכה 0 משמעה cluster מספר i תפוס וכאשר ערכה 1 ה-cluster פנוי. בדרך כלל ייכתב לסקטור הראשון מיד אחרי ה-VHD, אך מאפיין זה ניתן לשינוי בעת יצירת הדיסק.

לצורך מימוש בטבלה הגדרנו מבנה בשם DATtype שהינו בעצם bitset באורך 1600 – כמספר ה-cluster-ים בדיסק.

תיאור שדות המבנה:

sectorNr - זהו המספר הסידורי של הסקטור שמכיל את ה-DAT בדיסק.
dat - רצף של ביטים המסמלים איזה cluster תפוס כאמור לעיל.
emptyArea - כרגע לא בשימוש.

תיאור פונקציות המחלקה:

במבנה זה אין פונקציות.

DirEntry

מבנה זה מופיע בתחילת כל קובץ או תיקייה ומתאר את אורכו בעליו וכדו'.

תיאור שדות המבנה:

Filename - השם של הקובץ.
fileOwner - שם בעל הקובץ.
fileAddr - כתובת הסקטור הראשון של הקובץ.
crDate - תאריך יצירת הקובץ.
fileSize - גודל הקובץ, כמספר סקטורים 4 bytes.
eofRecNr - מיקום "רשומת" ה-end-of-file (המספר הסידורי של מיקומה מהתחלת הקובץ).
actualRecSize - אורך רשומה בפועל.
recFormat- אם מדובר על רשימה אז הוא מגדיר אם האורך קבוע או משתנה (F או V בהתאמה) ואם מדובר בתת תיקייה אז הערך D.
KeyOffset - של התחלת המפתח בתוך הרשומה.
KeySize - אורך המפתח, כמספר בתים.
keyType - טיפוס נתונים של ערך המפתח: "I" - int, "F" - float, "D" - double, "C" - char*)
entryStatus - שדה זה מעיד על מצב הכניסה הספציפית בתיקייה. המצב יכול להיות אחד מתוך שלושה:
0- כניסה ריקה - empty: הכניסה עדיין לא הייתה בשימוש מאז שבוצע format על הדיסק.

1- כניסה פעילה - active: הכניסה מייצגת קובץ קיים ופעיל.
2- כניסה לא פעילה - inactive: הכניסה מייצגת קובץ מחוק.

sLevel - רמת האבטחה של הקובץ.

תיאור פונקציות המחלקה:

DirEntry ()
בנאי ברירת מחדל שמאתחל את המחלקה בערכי ברירת מחדל.

DirEntry (...)
בנאי שמאתחל את המחלקה בערכים ספציפיים.

SectorDir

המבנה מתאר אחד משני הסקטורים אשר מכילים את כל המידע של ה DirEntry-ים (ראה עוד: RootDir).

ה-SectorDir כמעט ולא עומד בפני עצמו בפרויקט, אך הוא מהווה מרכיב חשוב ביצירת ה-RootDir

תיאור שדות המבנה:

sectorNr - זהו המספר הסידורי של הסקטור שמכיל את הסקטור בדיסק.
dirEntry - מערך של 14 DirEntries.
unUsed - שמור לשימוש עתידי.

תיאור פונקציות המחלקה:

במבנה זה אין פונקציות.

RootDir

המבנה מתאר את הcluster אשר מכיל את כל הDirEntry-ים בדיסק (ראה "DirEntry").

ה – RootDir למעשה ממפה את התיקיות, תתי תיקיות והקבצים ע"מ לאפשר אליהם גישה מהירה.
בפרויקט זה בחרנו שלא לממש תתי תיקיות אלא רק תיקייה ראשית המוגבלת ל-28 קבצים בלבד.

תיאור שדות המבנה:

lsbSector – מבנה מסוג Sector Dir הסקטור הראשון בcluster אשר מכיל 14 DirEntry-ים.
msbSector - מבנה מסוג Sector Dir הסקטור השני בcluster אשר גם הוא מכיל 14 DirEntry-ים.

תיאור פונקציות המחלקה:

במבנה זה אין פונקציות.

Sector

מבנה זה מתאר מבנה של כל סקטור בדיסק. גודלו של המבנה הינו 1024 בתים והוא מחייב כל מבנה הממלא סקטור במידע להיות בגודל זה.

> *עוד מחלקה שלא זכתה לכבוד הראוי לה. אמנם היא כמעט ואינה מוזכרת בקוד, אך היא מהווה את אבן היסוד הבסיסית ביותר המאפשרת את בניית הדיסק.*

תיאור שדות המבנה:

sectorNr - מספרו סידורי של הסקטור בדיסק.
rawData - השדה בו ישמרו הנתונים.

תיאור פונקציות המחלקה:

במבנה זה אין פונקציות.

FileHeader

מבנה הנמצא בסקטור הראשון של קובץ ומכיל את מאפייניו.

> *בהגדרת המבנה בחוברת הקורס הופיע שטח ריק בגודל 744 בתים. ע"מ לשפר את יעילות וביצועי המערכת בחרנו לנצל שטח זה ע"מ למפות את הרשומות בקובץ במבנה RecInfo. (חסרון השיטה נעוץ בכך שכעת כל קובץ מוגבל למספר מקסימלי של 46 רשומות).*

תיאור שדות המבנה:

sectorNr - מספר סידורי של הסקטור בדיסק.
fileDesc - העתק של כניסתו של הקובץ בתיקייה (העתק של ה-DirEntry).
fat - מייצג את הFAT.
RecInfo – ממפה בין מפתח רשומה למספר רשומה (ראה RecInfo).
emptyArea - שמור לשימוש עתידי, כרגע לא בשימוש.
sLevel – רמת האבטחה של הקובץ

תיאור פונקציות המחלקה:

FileHeader ()
בנאי ברירת מחדל שמאתחל את המחלקה בערכי ברירת מחדל.

FileHeader (...)
בנאי שמאתחל את המחלקה בערכים ספציפיים.

User

מבנה המייצג משתמש בדיסק.

מחלקה זו ונגזרותיה הינן תוספות שלנו לפרויקט.

תיאור שדות המבנה:

name – שם המשתמש.
password – סיסמא.
sLevel – רמת האבטחה של המשתמש.

תיאור פונקציות המחלקה:

User ()
בנאי ברירת מחדל שמאתחל את המחלקה בערכי ברירת מחדל.

User (...)
בנאי שמאתחל את המחלקה בערכים ספציפיים.

UserSec

מחלקה זה משמשת להזנת פרטי המשתמשים ורמות הגישה שלהם בקובץ (בתוך סקטור).

תיאור שדות המבנה:

sectorNr - מספר סידורי של הסקטור בדיסק.
NumOfUsers – מספר משתמשים (ערך ברירת מחדל 0).
users – מערך של (עד) 5 משתמשים.
RawData – שמור לשימוש עתידי.

תיאור פונקציות המחלקה:

UserSec ()
בנאי ברירת מחדל שמאתחל את המחלקה בערכי ברירת מחדל.

Disk

מייצגת את מבנה הדיסק. לב ליבו של הפרויקט.

תיאור שדות המבנה:

vhd – מייצג את ה-VHD.
dat - שדה שאליו ה- DAT יועבר בזמן ביצוע mount או ייווצר בזמן createDisk.

- rootdir שדה מסוג RootDir שאליו נתוני התיקייה הראשית של הדיסק המדומה יועברו בזמן ביצוע mount, או ייווצרו בזמן createDisk.
- mounted שדה המציין האם הדיסק מעוגן או לא.
– dskfl קובץ fstraem של המייצג את הדיסק הווירטואלי.
- currDiskSectorNr המספר הסידורי של הסקטור בדיסק שכרגע בחוצץ של קובץ מסוים.
– sign השדה sign שווה true כאשר משתמש מחובר (logged in) לדיסק, אחרת false.
– vhdUpdate האם ה-VHD עודכן במהלך ה-mount הנוכחי.
– rootDirUpdate האם ה-RootDir עודכן במהלך ה-mount הנוכחי.
– datUpdate האם ה-DAT עודכן במהלך ה-mount הנוכחי.
– currUser המשתמש (User) הנוכחי בדיסק.
– Users פרטי משתמשים (UserSec) בדיסק. נשמר בסקטור משלו בדיסק.
– usersUpdate האם עדכנו את הuserSec.

– lastErrorMessage מחזיר את הודעת השגיאה (exception) שזורקת אחת מפונקציות המחלקה (נועד לצורך סנכרון עם managed code).

תיאור פונקציות המחלקה:

() Disk
תפקידו של בנאי זה לאתחל את המחלקה עם ערכי ברירת מחדל.

(...) Disk
תפקידו של בנאי זה לאתחל את המחלקה עם ערכים ספציפיים.

() ~disk
תפקידה של הפונקציה הזאת להשמיד אובייקט מהסוג הזה באופן ברגע המתאים .

() createDisk
תפקידה של פונקציה זו ליצור דיסק מדומה.

() mountDisk
תפקידה של פונקציה זו לפתוח את הקובץ הממשמש את הדיסק המדומה.

() unMountDisk
תפקידה של פונקציה זו לעדכן את כל הסקטורים המכילים מידע מבני של הדיסק, לסגור את הקובץ של הדיסק המדומה ולתת לשדה mounted את הערך false .

() recreateDisk
תפקידה של פונקציה זו לאתחל מחדש את הדיסק המדומה .

() getDskfl
תפקידה של פונקציה זו להחזיר מצביע מסוג fstream לכתובת של dskfl שמייצג את הקובץ שמכיל את הדיסק המדומה.

() seekToSector
תפקידה של פונקציה זו להתמקם בסקטור המבוקש בדיסק המדומה.

() writeSector
תפקידה של פונקציה זו לכתוב מהחוצץ שאליו מצביע הפרמטר השני, לתוך הסקטור המבוקש(או הנוכחי) בדיסק המדומה.

() readSector
תפקידה של פונקציה זו לקרוא את הסקטור המבוקש(או הנוכחי) מהדיסק המדומה לתוך החוצץ שאליו מצביע הפרמטר השני.

addUser ()
הפונקציה מוסיפה משתמש לדיסק.

signIn ()
הפונקציה מאמתת את המשתמש (שם וסיסמא). מעדכנת את currUser.

signOut ()
תפקידה של פונקציה זו הוא לאפשר למשתמש לעשות logout (ע"מ להחליף משתמשים למשל).

removeUser ()
הפונקציה מוחקת משתמש מהדיסק (כאשר המשתמש אינו מחובר – מאפשר למשתמש למחוק את עצמו).

removeUserSigned ()
הפונקציה מוחקת משתמש מהדיסק (כאשר המשתמש מחובר).

changePassword ()
הפונקציה מאפשרת למשתמש לשנות את הסיסמא שלו.

שלב 1

בשלב זה מתוארת כתיבה של קבצים על דיסק ושימוש באמצעי האחסון ומיפוי המידע של הדיסק והקובץ.
לצורך כך הוגדרו הפונקציות המדמות פרמוט של דיסק, שמירה והרחבה של קובץ על הדיסק, מחיקת קובץ, ומתן מידע אודות המקום הפנוי בדיסק.
בפונקציות אלו נעשה שימוש המדמה את השימוש ב FAT וה- DAT בעת שימוש בפונקציות הכתיבה הקריאה והמחיקה של הקבצים.

כל הפונקציות בשלב זה שייכות למחלקה Disk.

format ()
תפקידה של פונקציה זו לעשות פרמוט לדיסק המדומה, הפונקציה תבצע:
- אתחול הDAT, כל הביטים של ה-DAT המייצגים cluster-ים של נתונים יקבלו ערך 1.
- אתחול התיקייה הראשית, כל כניסותיה של התיקייה הראשית יסומנו כריקות.

howmuchempty ()
תפקידה של פונקציה זו להחזיר את מספר סה"כ ה cluster -ים הפנויים בדיסק המדומה.

alloc ()
תפקידה של פונקציה זו לטפל בהקצאת שטח (לקובץ) בדיסק.

Allocextend ()
תפקידה של פונקציה זו לטפל בהוספת הקצאת שטח (לקובץ) בדיסק.

dealloc ()
תפקידה של פונקציה זו לשחרר שטח תפוס (ע"י קובץ מסוים).

first_fit ()
תפקידה של פונקציה זו להקצאות שטח לפי אלגוריתם "first fit", הפונקציה מחפשת את השטח הראשון שמספיק גדול לגודל המבוקש חותכת ומקצה אותו, אם היא לא מוצאת שטח מתאים היא לוקחת את השטח הראשון שפנוי ושולחת רקורסיבית את השטח הנותר.

best_fit ()
תפקידה של פונקציה זו להקצאות שטח לפי אלגוריתם "best fit", הפונקציה מחפשת שטח בגודל המדויק ומקצה אותו, אם היא לא מצאה היא מחפשת שטח גדול יותר, אם גם שטח גדול יותר היא לא מצאה אז היא לוקחת את השטח הכי גדול ושולחת רקורסיבית את הנותר.

worst_fit ()
תפקידה של פונקציה זו להקצאות שטח לפי אלגוריתם" worst fit", הפונקציה מחפשת את השטח שהכי פחות מתאים ,הפונקציה מחפשת תחילה את השטח הגדול ביותר, אם הוא מספיק גדול הפונקציה תחתוך ותקצה ,אחרת היא תקצה ותשלח ברקורסיה את הנותר.

שלב 2

בשלב זה מיושמות הפונקציות האחראיות על הקצאת השטח ליצירת קובץ או הרחבתו, ושחרור שטח מוקצה. בשלב זה הקובץ הוא רצף של סקטורים וכך אנו מתייחסים אליו במימוש הפעולות.

createfile ()
תפקידה של פונקציה זו הוא ליצור את התשתית לקובץ. כלומר, הקצאת השטח לצורך הקובץ, הזנת מאפייני הקובץ ושמירתם במקומות הנדרשים.

delfile ()
תפקידה של פונקציה זו הוא לשחרר שטח תפוס ע"י קובץ מסוים ומחיקתו באופן לוגי (ב-DAT).

extendfile ()
תפקידה של פונקציה זו הוא להרחיב את הקובץ. תפקידה הוא להקצות את השטח החדש ולעדכן את נתוני הקובץ במקומות הנדרשים.

שלב 3

בשלב זה מוגדרת ומיושמת מחלקת הFCB המאפשרת טיפול בקובץ מבחינה לוגית- כרצף של רשומות.

RecEntry

מבנה המייצג משתמש בדיסק.

מחלקה זו ונגזרותיה הינן תוספות שלנו לפרויקט. מטרת המחלקה היא ליצור צמדים של מפתח ומספר רשומה ע"מ לאפשר חיפוש מהיר יותר בקובץ.

תיאור שדות המבנה:

recNr – מספר רשומה.
key – מפתח של הרשומה.

תיאור פונקציות המחלקה:

RecEntry ()
בנאי ברירת מחדל שמאתחל את המחלקה בערכי ברירת מחדל.

RecEntry (...)
בנאי שמאתחל את המחלקה בערכים ספציפיים.

RecInfo

מבנה המייצג משתמש בדיסק.

מחלקה זו ונגזרותיה הינן תוספות שלנו לפרויקט. בדומה ל-DirEntry, המחלקה ממפה את הרשומות בכל קובץ, כך שניתן לעקוב בקלות אחר ברשומות בקובץ.

תיאור שדות המבנה:

records – מערך של 45 רשומות מסוג RecEntry.
size – הגודל ע"מ לדעת כמה רשומות יש בקובץ.

תיאור פונקציות המחלקה:

RecInfo ()
בנאי ברירת מחדל שמאתחל את המחלקה בערכי ברירת מחדל.

RecInfo (...)
בנאי שמאתחל את המחלקה בערכים ספציפיים.

FCB

המחלקה מתארת את הקובץ מבחינת התוכן שבו ז"א בצורה לוגית, כלומר, כרצף של רשומות. מחלקה זו תעזור לנהל את פעולות הקלט/פלט לפי רשומות, הקשורות לקובץ כלשהו בדיסק.

לכל קובץ שיפתח בדיסק ייווצר מופע חדש של FCB שיהיה אחראי לנהל אותו.

תיאור שדות המבנה:

d – מצביע לאובייקט מסוג Disk שמייצג דיסק מדומה.
Path – מספר המייצג את מיקומו של ה-DirEntry ב-RootDir של הדיסק.
fileDesc – העתק הfileDesk של הFileHeader של הקובץ.
FAT – הFAT של הקובץ.
Buffer – חוצץ המיועד לשמירת הסקטור שכרגע בשימוש.
currRecNr - מספר סידורי של הרשומה הנוכחית בתוך הקובץ.
currSecNr - מספר סידורי של הסקטור הנוכחי ,בתוך הקובץ.
currRecNrInBuff - מספר סידורי של הרשומה הנוכחית בתוך הסקטור שבbuffer.
MODE – משתנה המכיל את המידע כיצד הקובץ נפתח (לקריאה/כתיבה או גם וגם).
changed – משתנה בוליאני המראה האם הbuffer השתנה.
lock - משתנה בוליאני המציין האם הקובץ נעול לעריכה.
updateMode – משתנה בוליאני המגדיר האם המצב כעת הוא "מצב עריכה".

numOfRecords – מציין את מספר הרשומות הנוכחי בדיסק.
DAT – בדומה ל-DAT שבדיסק, ה-DAT פה הינו מערך של int-ים הממפה את המקומות הפנויים בקובץ.
loaded – משתנה בוליאני המציין אם ישנו קובץ הטעון ל-FCB.
recInfo – ראה RecInfo.

תיאור פונקציות המחלקה:

Default constractor ()
הפונקציה מאתחלת בערכי ברירת מחדל את המשתנים של המחלקה.

CTor ()
הפונקציה מקבלת מצביע מסוג דיסק ומשייכת אותו אל המשתנה המתאים בפונקציה בנוסף, הפונקציה מאתחלת את יתר המשתנים בערכי ברירת מחדל.

DTor ()
הפונקציה משחררת את כל המשתנים שהוקצו דינאמית ואת כל המצביעים.

Closefile ()
תפקידה של פונקציה זו לסגור את הקובץ הספציפי שמשוייך לFCB. הפונקציה כותבת פיזית את הבאפר ומעדכנת את כל מה שצריך לעדכן בFileHeader.

Flushfile ()
הפונקציה בודקת האם הנתונים שבבאפר השתנו ובהתאם לכך ,שומרת את המידה לסקטור המתאים.

readRec ()
תפקידה של פונקציה זו לקרוא רשומה מהבאפר אל המצביע שקיבלה כפרמטר.

writeRec ()
הפונקציה מקבלת מצביע אל הרשומה שמעוניינים להוסיף אל הקובץ הנוכחי והפונקציה מעדכנת זאת.

Seek ()
תפקידה של פונקציה זו למקם את המצביע לרשומה הנוכחית במיקום המבוקש במקרה הצורך תתבצע גם שמירה של הבאפר הנוכחי אל הדיסק וקריאה מחודשת של סקטור אחר לבאפר.

updateCancel ()
תפקידה של פונקציה זו לבטל את מצב הנעילה של הרשומה הנוצר בעקבות" קריאה לצורך עדכון.

deleteRec ()
תפקידה של פונקציה זו לבצע מחיקה לוגית לרשומה הנוכחית בקובץ.

updateRec ()
תפקידה של פונקציה זו לכתוב את הרשומה המעודכנת למיקום של הרשומה הנוכחית.

שלב 4

בשלב זה נִבְנָה הממשק הגראפי ואתו תוספות של פונקציות שאולצנו או החלטנו להוסיף, בשלב זה בוצעה גם ההתממשקות לתשתית שבנינו (שלבים 0-3) ובניית שני הפרויקטים המגשרים שפירטנו עליהם במבוא.

המשך תיאור המחלקה Disk

GetLastErrorMessage ()
תפקידה של פונקציה זו הוא להחזיר את השגיאה האחרונה שהתרחשה.

SetLastErrorMessage ()
תפקידה של פונקציה זו הוא לשמור את השגיאה האחרונה שהתרחשה.

getVolumeHeader ()
תפקידה של פונקציה זו הוא להחזיר את ה-VHD של הדיסק.

getDirEntry ()
תפקידה של פונקציה זו הוא להחזיר את ה-DirEntry מספר i (מתוך 28).

getFileHeader ()
תפקידה של פונקציה זו הוא להחזיר את ה-FileHeader של קובץ מסוים.

getDat ()
תפקידה של פונקציה זו הוא להחזיר את ה-DAT של קובץ מסוים.

נספח – מדריך למשתמש

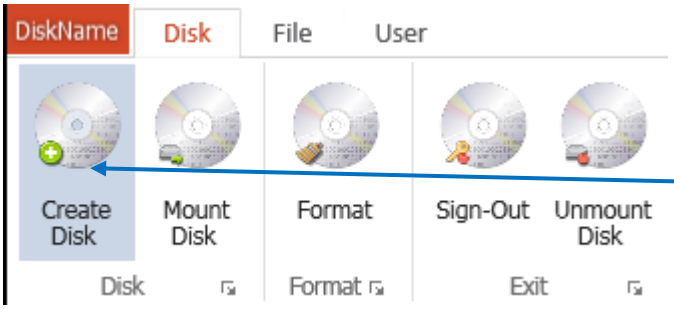
חלון פתיחה

בהרצת התוכנית ייפתח לנו חלון ובו שלושה תוויות: דיסק, מסמכים ומשתמשים. בנוסף משמאל שם הדיסק הנוכחי (כרגע אין דיסק בשימוש ומאותחל כ-DiskName).

כרטיסיית דיסק

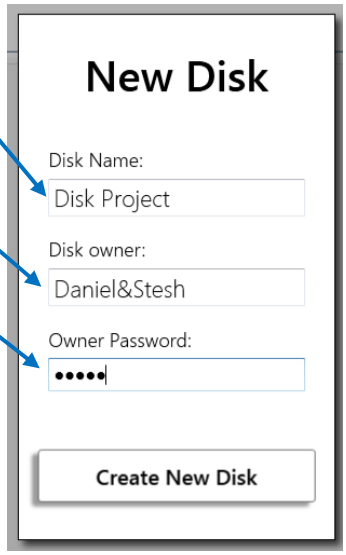
כרטיסיה זו נבחרת כברירת מחדל בהרצת התוכנית. כאן ניתן לבצע את כל הפעולות בדיסק (ליצור, לפרמט...).

לצורך הדוגמא: ניצור דיסק חדש.



ייפתח חלון חדש. נזין נתונים:

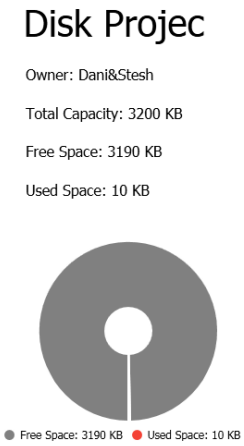
שם הדיסק שם בעל הדיסק וסיסמא.



ברגע שנלחץ על "צור דיסק" – תצוץ לנו חלונית חדשה להכנסת פרטי משתמש וסיסמא (log in): את פרטי המשתמש ונמשיך.

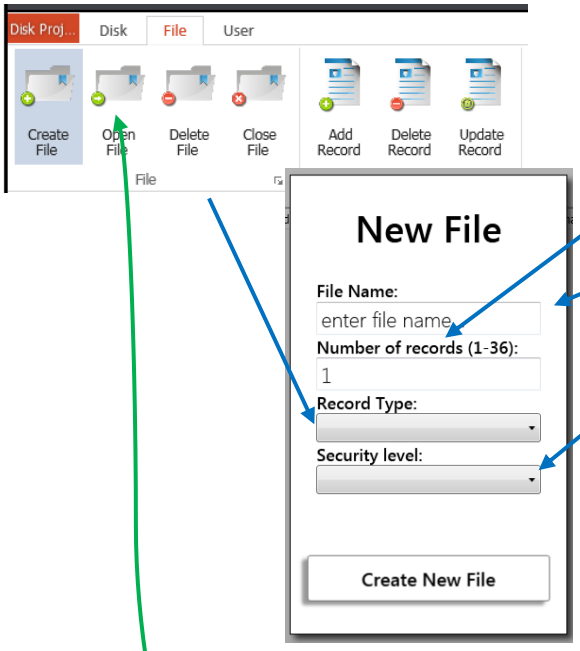
נאמת

כעת, יצרנו דיסק חדש.



כרטיסיית קבצים

נבחר בתווית "קבצים", וניצור קובץ חדש בדיסק:



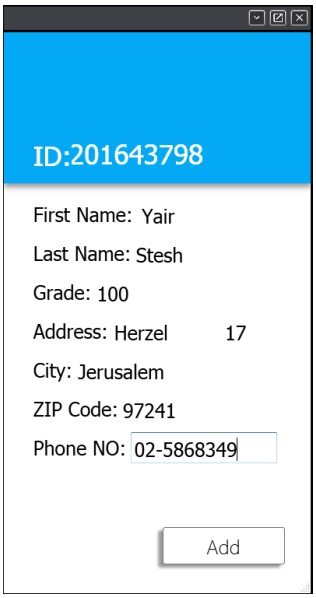
נזין שם קובץ, מס' רשומות, סוג רשומה ואת רמת האבטחה של הקובץ:

ניצור מספר קבצים בשביל הדוגמא.

FileName	FileOwner	FileAddr	CrDate	FileSize	EofRecNr	ActualRecSize	RecFormat	KeyOffset	KeySize	KeyType	EntryStatus	SLEVEL
file001	Dani&Stesh	6	07/11/16	4	22	88	F	0	10	s	1	4
file003	Dani&Stesh	14	07/11/16	2	11	88	F	0	10	s	1	4
file002	Dani&Stesh	10	07/11/16	4	33	88	F	0	10	s	1	3
GuideFile	Dani&Stesh	16	07/11/16	2	11	88	F	0	10	s	1	4

עכשיו נרצה להוסיף רשומות לאותו קובץ, נבחר בו בטבלה, ונלחץ: "פתח קובץ".

הקובץ נפתח, נוכל לראות מידע אודותיו ולהוסיף רשומות ע"י כפתור: "הוסף רשומה".



נזין פרטים: מפתח רשומה, שם פרטי, משפחה, ציון, רחוב, מס' בית, עיר, מיקוד וטלפון.

נוכל לעיין ברשומות שבקובץ בטבלה מימין.

כמו-כן, נוכל לבחור בטבלה ברשומה מסוימת ולמחוק/לשנות אותו.

> *הערה זו תקפה גם לגבי הקבצים, נבחר את הקובץ בטבלה ונמחק/נשנה אותו.*

בפעמים הבאות, נוכל להעלות את הדיסק (המדומה) ע"י בחירה בכרטיסיית "דיסק" בכפתור: "mount Disk" וכך נוכל להמשיך מאיפה שהפסקנו.

סיכום כללי והצעות לעתיד

במהלך הפרויקט למדנו על מבנה הדיסק. בסמסטר הקודם (קורס ארגון וניהול קבצים) למדנו גם על הדיסק ומבנהו אך בפרויקט העמקנו את הידע והבנו אותו ברובד המעשי.

עצם העבודה על הפרויקט אילצה אותנו להבין איך בנוי כל מבנה בדיסק, מדוע צריך את כל השדות של המבנים ואת הפונקציות שכל מבנה מממש. למדנו גם כיצד מתבצעת קריאה וכתיבה של רשומות וסקטורים בדיסק. העמקנו את ההבנה והצורך בFCB וכיצד הוא עובד.

בשביל שנוכל לשלב בין כל שלב ושלב בפרויקט נדרשנו להעמיק את ההבנה של כל שלב בפני עצמו ואת הצורך ההכרחי שלו לשלבים הבאים, למשל בשביל לממשק בין ה FCB לדיסק עצמו נדרשנו להבין את הקשר בניהם וכיצד משתמש הFCB בדיסק לצורך פעולותיו.

דבר נוסף שיישמנו אותו קשור בקורס מבוא לאבטחת מידע שלמדנו הסמסטר. לכל קובץ ודיסק הגדרנו רמת אבטחה ע"פ המודל של RBAC. כלומר, לכל קובץ יש רמת אבטחה שמוגדרת ע"פ תפקידו וההרשאות הם בהתאם, בצורה היררכית.

דבר נוסף, לצורך חלוקת המשימות, התקדמות במקביל (לעיתים על אותו קובץ עצמו), והצורך לשמור על סנכרון תמידי – השתמשנו ב**visual studio team server**. זה היה תהליך מרתק ומלמד. זהו כלי יעיל מאוד לעבודה בצוות: התכנון, חלוקת המשימות, שלבי התקדמות, גיבוי, מיזוג קטעי קוד שונים ובחירה בין שינויים. אנו ממליצים שמי שעושה פרויקט ישתמש בכלי הזה (או כלי כדוגמתו...).

הצעה נוספת, החלק שבו נדרשנו לעבור מC++ לC# היה מעט מיותר. אמנם למדנו בעבר מימוש גרפי בWPF של .NET. אבל יכולנו לכתוב את התכנית כולה בC# או לחילופין בC++ ולממש עם ממשק גרפי מתאים.