תרגיל בית 1

<u>מועד הגשת התרגיל:</u> עד יום שבת 3/11/18 בשעה 23:55. לא תהינה דחיות

מטרת התרגיל

- ריענון השימוש בסביבת הפיתוח Microsoft Visual Studio.
- ריענון של תכנות בשפת C עם פעולות קלט/פלט והקצאה דינאמית של זיכרון.
- תרגול תכנות נכון: קוד קריא, חלוקה למודולים, שימוש בקבועים, תיעוד ועוד.

הגשה

צורת ההגשה מפורטת במסמך "הנחיות להגשת תרגילי בית – תשע"ט" שבאתר המודל. אנה הקפידו למלא אחר ההוראות.

הגישו פרויקט מלא, כולל קבצי פרויקט (*.sln, *.vcxproj, *.vcxproj.filters) של Visual Studio 2017, באופן שיאפשר לבודק התרגילים לפתוח את הפרויקט על ידי לחיצה כפולה על קובץ ה-solution ולקמפל את הפרויקט ללא התראות או שגיאות.

הגישו בנוסף את תיקיית ה-Debug עם הקובץ ה-exe.

דגשים

בתרגיל הראשון ננחה אתכם איך לחלק את הקוד למודולים ואף נגדיר את חלק מהפונקציות. בתרגילים הבאים תידרשו לעשות זאת בעצמכם.

הקפידו על קוד קריא ומתועד.

עבדו באיטרציות. בדקו את הקוד שכתבתם לפחות בסוף כל סעיף.

זכרו להשתמש בכלי הדיבוג שה-IDE מספק.

הפורום עומד לשירותכם. אנו מעודדים אתכם לנסות תחילה לחפש תשובות באינטרנט, כאשר מדובר בשאלות תכנות כלליות.

בהצלחה!

סקירה כללית

חברת התעופה "עסקי אוויר" בקשה מכם לבנות עבורה מערכת לניהול טיסות.

אנו מעוניינים לבנות תוכנית שתקבל כקלט את פרטי הטייסים הזמינים בחברה, ואת היעדים המתוכננים.
התוכנית תקצה את המטוסים והטייסים הרלוונטיים לכל טיסה. ראו פירוט והרחבות בסעיפים הבאים.

משימות תכנות

כעת נמשיך לכתיבת התוכנה מודול אחר מודול. זכרו לבדוק את עצמכם לאחר כל סעיף, ולא להמתין להשלמת התוכנה.

באפשרותכם לכתוב פונקציות, משתנים, קבועים וטיפוסים נוספים, אם הם מסייעים לכם בכתיבת הקוד.
בקוד הסופי השאירו רק את חתיכות הקוד שבאות לידי שימוש, כלומר אסור שתופענה במסך או בקובץ הפלט הודעות נוספות.

פרויקט חדש

צרו פרויקט חדש. קראו ל-project ול-solution בשם ex1. זה יהיה גם שם ה-executable.
לקובץ הראשי קראו main.c.

מודול airplane_db

מודול זה מגדיר את בסיס הנתונים של המטוסים הקיימים בחברה.
צרו קבצים חדשים עבור המודול: airplane_db.c ו-airplane_db.h.

סעיף 1

הגדירו טיפוס חדש מסוג "דגם מטוס". הטיפוס הוא מבנה (struct), שמכיל את השדות הבאים:

שם שדה	תיאור שדה
דגם המטוס	מחרוזת בעלת 3 תווי ASCII
יעדים אליהם המטוס טס	מערך של מחרוזות

המודול מחצין את ההגדרה הזו, כלומר היא נמצאת ב-header.

כעת, הגדירו קבוע בדמות מערך "דגמי מטוסים" (הטיפוס שהגדרנו לפני רגע). המערך יראה כך:

Type	Destinations
737	Larnaca, Athens, Budapest, Zurich, London, Paris, Rome
747	London, New York, Bangkok
787	London, New York, Los Angeles, Hong Kong, Miami

קבוע זה ישמש כקבוע גלובלי המוגדר ב-airplane_db.c
שימו לב שישנם מספר יעדים ששני מטוסים יכולים להגיע אליהם!

סעיף 2

הגדירו פונקציה בשם GetAirplaneType. הפונקציה מבצעת שליפה מה-db – הפונקציה מקבלת שם של יעד (מחרוזת), ופולטת מצביע לדגם ב-db.

פרמטרי קלט: שם יעד
פרמטרי פלט: מצביע לדגם ב-db

החזרה: הפונקציה מחזירה 0 עבור הצלחה, ערך שלילי כרצונכם עבור כישלון.
אם לא נמצא ב-db יעד בעל השם שהתקבל, או אם התקבל NULL pointer, או אם קרתה תקלה אחרת, הפונקציה תחזיר ערך שלילי.

רמז לסעיף 2: השארנו לכם את החופש להחליט איזה דגם יוחזר במקרה של יעד שעבורו ישנם שניים או שלושה דגמים היכולים לטוס אליו. בהמשך התרגיל, תצטרכו לבחור *עבור כל יעד*, את המטוס הצעיר ביותר שמגיע לשם. יש כמה דרכים לבצע את זה. הנה אפשרות אחת:

החזירו את הדגם הראשון הרלוונטי. למשל, עבור London - יוחזר מצביע ל-"737". לאחר שבסעיף 5 תמצאו את המטוס הצעיר ביותר מדגם זה, שמרו אותו בצד. כעת, קראו שוב ל-GetAirplaneType . כיצד נוכל לקבל הפעם "747" עבור הקלט London, ולאחר מכן "787" כדי למצוא את המטוס הצעיר מבין כל הדגמים?
ניתן להגדיר מסכה חיצונית (למשל מערך גלובלי בעל 3 איברים שמאותחלים לאפס):

```
int AirplanesFound[NUM_OF_OPTIONS]={0,0,0};
```

כל איבר כאן מייצג דגם אחר. במהלך הקריאה הראשונה לפונקציה, תהיה התאמה ב-"737". לכן, קבעו 1 במערך העזר:

{1,0,0}

כדי שתהיה התאמה ליעד מסויים בפונקציה שלנו, צריכים כמובן שהמחרוזת תתאים, אבל הוסיפו תנאי גם שהאיבר הרלוונטי במערך העזר יהיה 0. כלומר, בקריאה השנייה לפונקציה (עם הקלט London) – כל ההתאמות של "737" יכשלו, כי אנו מבצעים and עם האיבר הראשון במערך העזר. משם נמשיך לחפש ביעדים של "747", ונקבל התאמה, כי האיבר האמצעי במערך שווה ל-0. לאחר הקריאה השנייה, המערך צריך להיות:

{1,1,0}

לכן בקריאה השלישית, רק ה-"787" יוכל לקבל התאמה.
להבדיל, נניח שאנו מחפשים את היעד Athens. נמצא אותו ב-"737", ולכן אחרי הקריאה הראשונה הווקטור שלנו יראה:

{1,0,0}

בקריאה השנייה, "737" חסום ע"י המערך, ולכן האופציות היחידות הן "747" ו-"787". באף אחת מהן אין Athens, ולכן הפונקציה תחזיר מספר שלילי, ואין צורך לקרוא לה שוב.

סעיף 3

הגדירו טיפוס חדש מסוג "מטוס". הטיפוס הוא מבנה (struct), שמכיל את השדות הבאים:

שם שדה	תיאור שדה
שם המטוס	מחרוזת
דגם המטוס	מחרוזת בעלת 3 תווי ASCII
גיל	מספר חיובי מסוג float המייצג את מספר השנים מאז בנייתו
המטוס הבא	מצביע לאיבר מסוג "מטוס". ישמש לבניית רשימה מקושרת.

המודול מחצין גם את ההגדרה הזו, כלומר היא נמצאת ב-header.

סעיף 4

הגדירו פונקציה בשם CreateAirplaneList

הפונקציה יוצרת רשימה מקושרת של המטוסים הבאים בעזרת הקצאת זיכרון דינאמית:

Airplane Name	Type	Age
Beit-Shean	737	5
Ashkelon	737	10.25
Hadera	737	3
Kineret	737	7.5
Nahariya	737	1
Tel-Aviv	747	20
Haifa	747	15
Jerusalem	747	17
Ashdod	787	1
Bat Yam	787	1.5
Rehovot	787	0.5

פרמטרי קלט: מצביע מסוג "מטוס". (מותר להשתמש במצביע למצביע כדי לשנות את המצביע בפונקציה השולחת)

פרמטרי פלט: מצביע לראש הרשימה (מומלץ שזה יהיה למעשה פרמטר אחד שמשמש גם כקלט וגם כפלט)

החזרה: הפונקציה מחזירה 0 עבור הצלחה, ערך שלילי כרצונכם עבור כישלון.

מותר לאיברים ברשימה המקושרת להופיע בכל סדר שתרצו.

סעיף 5

הגדירו פונקציה בשם GetAirplane. הפונקציה מקבלת דגם של מטוס (מחרוזת), ומצביע לראש רשימת המטוסים, ופולטת מצביע למטוס ברשימה - בעל הגיל <u>*הצעיר*</u> ביותר מאותו הדגם.

פרמטרי קלט: דגם מטוס (מחרוזת), מצביע לראש רשימת המטוסים.

פרמטרי פלט: מצביע למטוס ברשימה

החזרה: הפונקציה מחזירה 0 עבור הצלחה, ערך שלילי כרצונכם עבור כישלון.

אם לא נמצא ברשימה מטוס בעל הדגם שהתקבל, או אם התקבל NULL pointer, או אם קרתה תקלה אחרת, הפונקציה תחזיר ערך שלילי.

סעיף 6

הגדירו פונקציה בשם DeleteAirplane. הפונקציה מקבלת מצביע למטוס ברשימת המטוסים ומוחקת אותו מהרשימה.

פרמטרי קלט: ***מצביע למצביע לראש הרשימה***, ומצביע למטוס שאותו רוצים למחוק.

פרמטרי פלט: המצביע למצביע מהקלט, ישמש גם כפלט. אם מוחקים את המטוס הראשון ברשימה, נרצה **שהמצביע לראש הרשימה** (מהפונקציה השולחת), יצביע כעת למטוס השני, או ל-NULL אם אין יותר מטוסים.

סעיף 7

הגדירו פונקציה בשם ClearAirplaneList

פרמטרי קלט: מצביע לראש רשימת המטוסים

פרמטרי פלט: אין

החזרה: הפונקציה אינה מחזירה דבר

פונקציה זו עוברת איבר-איבר ומשחררת את הזכרון שהוקצה באופן דינמי.

מודול pilots

המודל הזה אחראי לטיפול בטייסי החברה.
צרו קבצים חדשים עבור המודול: pilots.c ו-pilots.h .

סעיף 8

הגדירו טיפוס חדש מסוג טייס. הטיפוס הוא מבנה (struct), שמכיל את השדות הבאים:

שם שדה	תיאור שדה
שם הטייס	מחרוזת באורך לא ידוע. כיצד נתמודד עם זה?
דגם מטוס	הדגם שהטייס/טייסת מוסמכים להטיס. מחרוזת בעלת 3 תווי ASCII
מספר שעות טיסה	מספר שעות הטיסה עד כה החודש. מספר שלם (חיובי)
דרגה	מחרוזת בעלת הערכים Captain או First Officer
הטייס הבא	מבציע לאיבר מסוג "טייס". ישמש לבניית רשימה מקושרת.

המודול מחצין את ההגדרה הזו, כלומר היא נמצאת ב-header.

סעיף 9

הגדירו פונקציה בשם GetPilots. פונקציה זו פותחת את הקובץ ששמו (נתיב) מועבר כארגומנט (מחרוזת) וקוראת כל שורה מתוכו. כל שורה בקובץ הינה בפורמט הבא: **Pilot Name, Airplane Type, Hours Flown, Rank**

לאחר דרגת הטייס ישנה ירידת שורה ללא רווח. לאחר הערכים האחרים באותה שורה, יופיע פסיק ורווח בודד. למשל:

Avi Ron, 737, 17, Captain

הפונקציה תבנה רשימה מקושרת שבה כל איבר מייצג טייס/טייסת.
בסיום הפונקציה, יש לסגור את קובץ הטקסט.

פרמטרי קלט: נתיב לקובץ
פרמטרי פלט: מצביע לראש הרשימה
החזרה: הפונקציה מחזירה 0 עבור הצלחה, ערך שלילי כרצונכם עבור כישלון.
אם לא נמצא קובץ כזה, או אם התקבל NULL pointer, או אם קרתה תקלה אחרת, הפונקציה תחזיר ערך שלילי.

סעיף 10

הגדירו פונקציה בשם DeletePilots. הפונקציה מקבלת מצביע לטייס/ת ברשימת הטייסים ומוחקת אותם מהרשימה.
פרמטרי קלט: ***מצביע למצביע לראש הרשימה***, ומצביע לטייס/ת שרוצים למחוק.
פרמטרי פלט: המצביע למצביע מהקלט, ישמש גם כפלט. אם מוחקים את ראש הרשימה, נרצה **שהמצביע לראש הרשימה** (מהפונקציה השולחת), יצביע כעת לטייס השני, או ל-NULL אם אין יותר טייסים.
החזרה: כלום

סעיף 11

הגדירו פונקציה בשם ClearPilotList
פרמטרי קלט: מצביע לראש רשימת הטייסים
פרמטרי פלט: אין
החזרה: הפונקציה אינה מחזירה דבר
פונקציה זו עוברת איבר-איבר ומשחררת את הזכרון שהוקצה באופן דינמי.

Main – הקובץ הראשי

קראו לקובץ main.c.

התוכנית שלכם תקבל 3 ארגומנטים (מלבד שם התוכנית שמתקבל אוטומטית): נתיב לשם של קובץ טייסים, נתיב לשם של קובץ יעדים, נתיב לשם של תוכנית הטיסות. כלומר, הארגומנטים (כולל שם התוכנית) יכולים להראות כך:

ex1.exe Pilots.txt Destinations.txt Plan.txt

שימו לב ששני קבצי הטקסט הראשונים מהווים קלט, והקובץ האחרון הינו פלט התוכנית. השמות של קבצים אלו יכולים להיות כמובן שונים ממה שמופיע כאן.

רשימת היעדים הינו קובץ טקסט שבו מופיעים (חלק מ- או כל) היעדים אליהם החברה טסה, המייצגים את תוכנית הטיסות להיום. כל יעד יופיע בשורה חדשה. לאחר כל יעד יש ירידת שורה. למשל:

Larnaca

Athens

New York

סעיף 12

התוכנית שלכם תעבור יעד-יעד, ותמצא לכל יעד (לפי הסדר) את המטוס הצעיר ביותר שמגיע ליעד זה. לאחר מכן, היא תמצא את זוג הטייסים (קפטן וקצין ראשון) בעלי מספר שעות הטיסה הנמוך ביותר המוסמכים לטוס על מטוס זה. התוכנית תדפיס את שם היעד, לאחר מכן את שם המטוס, את שם הקפטן, ואת שם הקצין הראשון (הפלטים בתוך כל שורה מופרדים בפסיק ורווח). לאחר שמו של הקצין הראשון תופיע ירידת שורה. ההדפסות יבוצעו לקובץ שנתיבו התקבל כארגומנט האחרון לתוכנית. דוגמה לפלט:

Larnaca, Nahariya, Avi Ron, Dani Matos

הנחות והנחיות:

- אם קרתה תקלה במהלך ריצת התוכנית, ודאו שקובץ הפלט יכיל את ההודעה הבאה **<u>בלבד</u>**:
  "An error occurred during execution, couldn't complete the task!"

במקרה זה כלולים יעדים בעייתים, מחסור בטייסים (למשל אם אין קפטן/קצין ראשון מתאימים) וכו'...

- יתכן וישנן 2 טיסות לאותו היעד. במקרה זה, ההדפסה הראשונה לאותו היעד תקבל את המטוס הצעיר ביותר (ואת הטייסים בעלי שעות הטיסה הנמוכים ביותר לאותו דגם).
- השוו את קובץ הפלט שלכם לדוגמה שהועלתה למודל, וודאו שעבור קבצי הפלט בדוגמה, קיבלתם בדיוק את אותה התוצאה.
- זכרו לשחרר זכרון דינאמי שהקצתם במהלך התוכנית.
- השתמשו במודולים מהסעיפים הקודמים.
- ניתן להניח שהקלט תקין. למשל, דרגות הטייסים תמיד תהיינה Captain או First Officer. עם זאת, עלולים להופיע יעדים שאין לנו מטוסים שיכולים להגיע אליהם, או שאין מספיק טייסים להשלמת תוכנית הטיסה.
- דאגו לחלק את קובץ ה-main גם כן לפונקציות בגודל הגיוני. פונקציית ה-main צריכה בעיקר לקרוא לפונקציות נוספות.