תרגיל בית 1

מועד הגשת התרגיל: עד יום שבת 17/11/19 בשעה 23:55. לא תהינה דחיות

בודק אחראי: אורי ברכה uribracha@mail.tau.ac.il

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.
- היכרות ראשונית ובסיסית עם יצירת תהליכים ו-WinAPI.

הגשה

צורת ההגשה מפורטת במסמך "הנחיות להגשת תרגילי בית – תש"ף" 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 ה-Exe-ים.

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

מטרת התרגיל היא יצירת היכרות והבנה בסיסית של יצירת תהליכים (Processes) לצד ריענון כתיבת קוד בשפת C.

הנחות והנחיות

- הימנעו משימוש במספרי קסם ומחרוזות מפורשות בקוד עצמו – רכזו את כל זה בקובץ ה-HardCodedData.h שיכיל את כל ה-define#-ים שלכם.
- בכל מקרה שלא מוגדרת התנהגות נדרשת אתם רשאים לבחור בכל התנהגות **סבירה**. התנהגות סבירה כוללת בתוכה סיום אלגנטי של התוכנית (לא לקרוס) תוך הדפסת הודעת שגיאה מתאימה.
- בכל הפרויקטים שניצור בתרגיל זה חייב להיות קובץ main.c שמהווה את נקודת הכניסה של התכנית – קרי את המקום שממנו התכנית מתחילה לרוץ.
- זכרו לשחרר זכרון דינאמי שהקצאתם במהלך התוכנית.
- זכרו לסגור HANDLE-ים.
- דאגו לחלק את קובץ ה-main גם כן לפונקציות בגודל הגיוני. פונקציית ה-main צריכה בעיקר לקרוא לפונקציות נוספות.
- **חובה** לקרוא על כל אחת מהפונקציות שמתוארות בהמשך באמצעות הקישורים/חיפוש בגוגל לצורך הבנה מיטבית שלהן לתרגיל זה ולטובת התרגילים בהמשך הקורס.

טיפ של אלופים: מומלץ **מאוד** לעבוד עם שירות Version Control כלשהו למשל כמו GitHub על מנת לשמור את ההתקדמות שלכם וזאת משום שהתרגיל הוא איטרטיבי. ביצוע commit לאחר השלמת מדרגה בתרגיל יעזור לכם לעקוב אחר ההתקדמות שלכם, לזהות בעיות ויאפשר לכם לחזור לנקודה האחרונה שבה הקוד שלכם עבד במקרה שהסתבכתם ואתם לא מצליחים למצוא את הבעיה. למי מכם שמעולם לא השתמש ב-Git ו-GitHub עדיף מאוחר מאשר אף פעם.

רקע כללי

בתרגיל זה תכתבו שתי תכניות שונות, כלומר, התרגיל יכיל שני פרויקטים שונים, לכל אחד מהם יהיה קובץ main.c משלו וכל אחד מהם יקומפל ל-executable משלו. המטרה היא שתכנית אחת תעשה שימוש בתכנית השנייה.

לתהליך שמריץ את התכנית שמריצה את התכנית השנייה נקרא תהליך אבא או תהליך אב ולתהליך שהאבא יוצא (ויריץ את התכנית השנייה) נקרא תהליך בן (Son).

הערה: התרגיל נכתב בלשון זכר אך פונה לאבות, אימהות, בנים ובנות.

בנוסף, תהליך האב יבצע כתיבה לקובץ כפי שיפורט בהמשך בעוד תהליך הבן יחזיר תוצאה באמצעות ה-exit code שלו (הערך שמוחזר מפונקציית ה-main שבקובץ ה-main.c שלו).

מדרגה 1

תהליך הוא מופע מסוים של תוכנה ולפיכך אותה תוכנה יכולה לרוץ בכמה תהליכים בו-זמנית.
לפעמים כאשר אנחנו כותבים קוד נרצה שחלקים שונים שלו יבוצעו על ידי תהליכים שונים מסיבות רבות כגון מקבול, הרשאות וכו'.

בסעיף זה נכתוב תכנית קטנה שמקבלת כמחרוזת ביטוי מתמטי פשוט דרך ה-Command Line בצורה הבאה:

"X+Y"

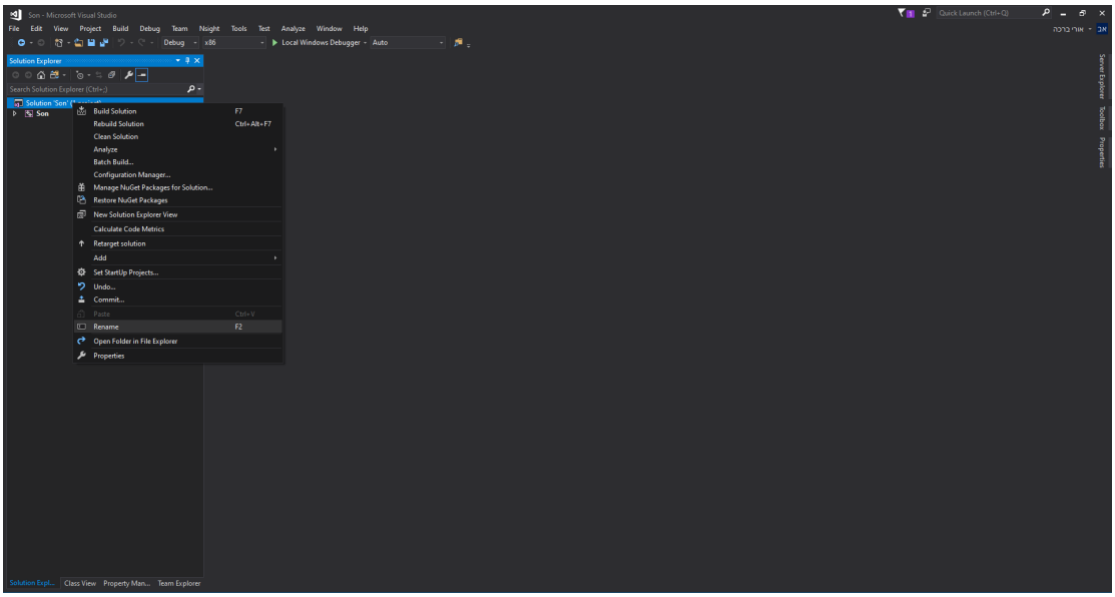
"X*Y"

התכנית תפרסר את הקלט ותחזיר את תוצאת הפעולה כ-exit code של התהליך. ניתן להניח ש-X ו-Y הם מספרים שלמים וחיוביים וכן שהפעולת היחידות הן פעולות חיבור וכפל.
ניתן להניח שהקלט תקין אבל יש לבדוק את הערך החזרה של כל פונקציה שאתם משתמשים בה ושיכולה להיכשל ולהתמודד עם זה כפי שמוסבר בהנחיות בתחילת התרגיל וכן ניתן להניח שאורך הקלט לא יהיה גדול מ-32 תווים (כולל בתוכו את ה-null terminator שבסוף הקלט).

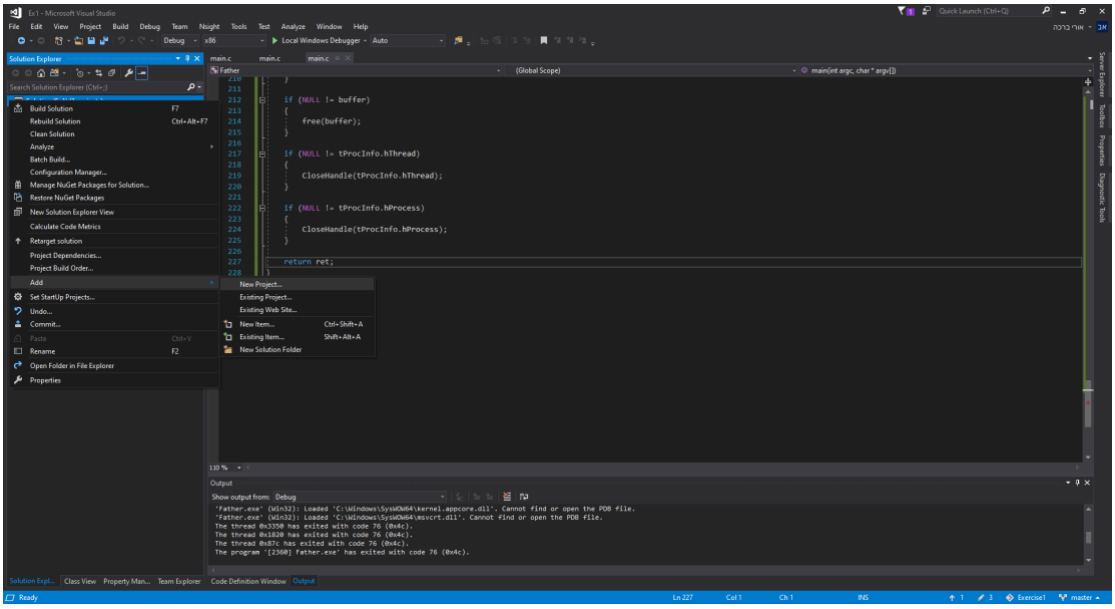
בתור התחלה פתחו Solution חדש ובתוכו פרויקט בשם Son. בתוך פרויקט זה ממשו את התכנית הנ"ל.

מדרגה 2

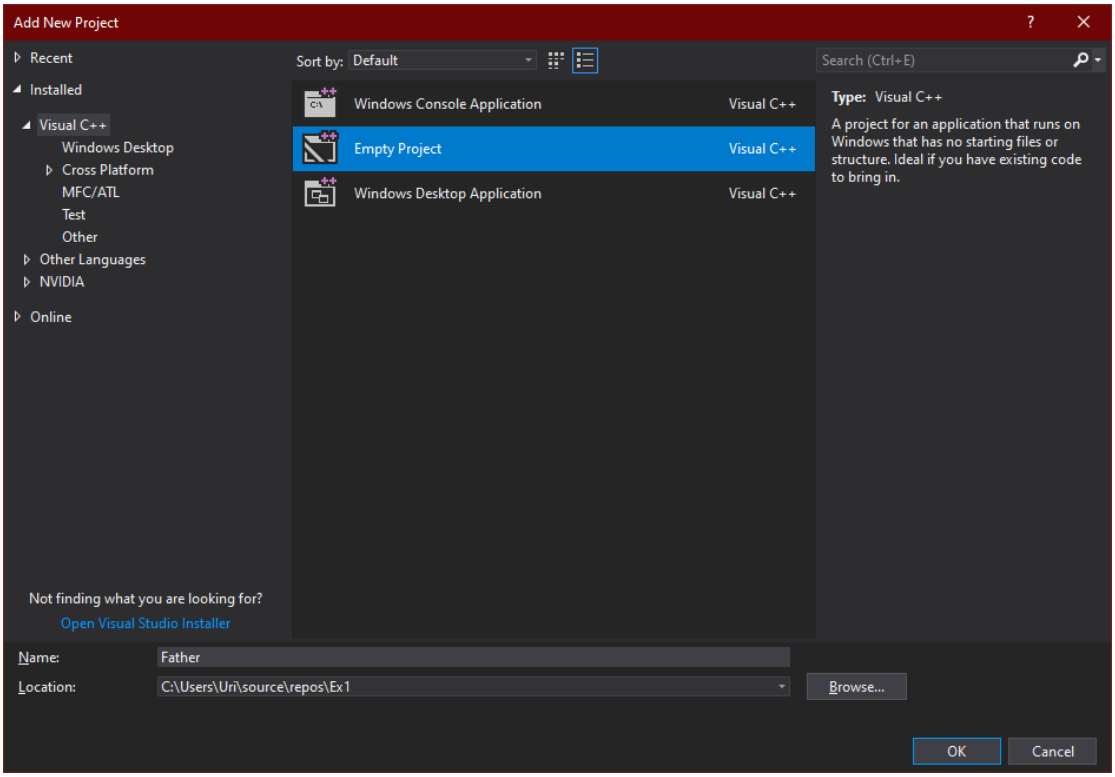
שנו את שם ה-Solution ל-Ex1:



כעת נוסיף פרויקט נוסף ל-Solution שלנו:



קראו לפרויקט החדש Father:



מדרגה 3

בפרויקט החדש כתבו תכנית שתקבל כקלט ביטוי מתמטי בפורמט הבא דרך ה-Command Line:

((1+3)*(4+(10+5)))

כלומר, הביטוי המתמטי מורכב מאוסף של ביטויים פשוטים. ניתן להניח שכל פעולה (חיבור וכפל) מתוחמת על ידי סוגריים, אין רווחים, בין כל שני מספרים תהיה פעולת חיבור או כפל כלומר **לא** תקבלו ביטוי מהצורה הבאה:

((1+3)*4+10+5)

במילים אחרות אתם יכולים להיעזר בסוגריים כדי לקבוע את סדר הפעולות. אתם יכולים להניח שהקלט שתקבלו יהיה תקין אך בכל מקרה של שגיאה אחרת שלא קשורה לקלט אתם **חייבים** להדפיס הודעת שגיאה אינפורמטיבית למסך ולסיים את ריצת התכנית באלגנטיות – בלי לקרוס. בנוסף, ניתן להניח שאורך הקלט לא יהיה גדול מ-256 תווים (כולל בתוכו את ה-null terminator שבסוף הקלט).

התכנית תיקח את הביטוי ותחשב אותו על ידי יצירת תהליכי בנים (כפי שמומשו בסעיף הראשון) והעברת ביטויים פשוטים יותר לבנים דרך ה-Command Line של הבנים בעת היצירה שלהם. על התכנית לחכות שהבן יסיים ולקבל את התוצאה ממנו. לאחר קבלת תוצאה מתהליך הבן תהליך האב יבנה מחדש את הביטוי (כמחרוזת) וידפיס את השינוי לקובץ בשם **Computation.txt** שיכיל את כל שלבי החישוב.

עבור הדוגמה שלמעלה התוכן של Computation.txt יהיה:

((1+3)*(4+(10+5)))
(4*(4+(10+5)))
(4*(4+15))
(4*19)
76

התוכן שלכם **חייב** להיראות **בדיוק** כמו בדוגמה הזאת (שימו לב לסדר הטיפול בביטויים הפשוטים, מטפלים באופן חמדני – ברגע שמוצאים ביטוי פשוט שאפשר לשלוח לבן שולחים אותו ומדפיסים את השלב לקובץ).

זכרו לדרוס את הקובץ בכל הרצה אם הוא כבר קיים וכן לסגור אותו לאחר הכתיבה.

חשוב: זכרו לקרוא את ההנחיות וההנחות בתחילת התרגיל!