האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע`ש רחל וסלים בנין

סדנאות תכנות בשפת C ו־++C - 67312
++C - תרגיל 2 (גרסה 2)

תאריך ההגשה של התרגיל והבוחן התיאורטי: יום רביעי, ה־8 בינואר, 2020 - עד השעה 23:55;

הגשה מאוחרת (בהפחתת 10 נקודות): יום חמישי, ה־9 בינואר, 2020 - עד השעה 23:55.

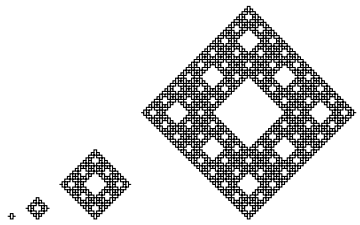
נושאי התרגיל: תכנות מונחה עצמים ו־STL.

1 רקע

בתרגיל זה נתנסה בתכונות הירושה של ++C על ידי כתיבת תוכנית המדפיסה עצים פרקטלים.

2 עצים פרקטלים

פרקטל זו צורה גאומטרית המורכבת מעותקים מוקטנים של עצמה. כלומר, מדובר בצורה שחוזרת על עצמה באופן רקורסיבי, כך שאם נבחן חלק מסויים של הפרקטל תחת "זכוכית מגדלת", נגלה שהמראה יהיה זהה בצורתו לפרקטל המקורי. כך, גם אם נבחן את אותו החלק עצמו - נמשיך לראות זאת הלאה והלאה, עד לרמה הראשונה. לפני המשך ההסבר, נראה דוגמה כדי לחדד את האינטואיציה. הפרקטל הבא נקרא Sierpinski carpet, והאיטירציות 1 עד 4 שלו, משמאל לימין, נראות כך:



איור 1: Sierpinski Carpet Orders 1 - 4

הבחינו שככול שאנו מתקדמים באיטירציות, הצורה חוזרת על עצמה בצורה רקורסיבית עוד ועוד. עתה, מהצגנו את הדוגמה - נמשיך. ישנן דוגמאות רבות נוספות לעצים פרקטלים, וחלק מהן נראה בתרגיל זה. לפרקטלים ישנם תכונות מתמטיות המאפיינות אותם. עם זאת, אנו לא נבחן בתרגיל זה את הפן המתמטי של פרקטלים, ונסתפק בכך שנציין שמרבית הפרקטלים מבוססים - כאמור - על התנהגות רקורסיבית. כמו כן, נגדיר את **הממד של הפרקטל,** אשר מסמל את דרגת האיטירציה. למשל, ב־Sierpinski Carpet שלעיל, הדרגה של האיור הימני ביותר היא 4, בעוד הדרגה של האיור השמאלי ביותר היא 1.

3 ציור פרקטלים באמצעות תווי ASCII

בתרגיל זה נתרגל תכנות מונחה עצמים, בדגש על פולימורפיזם ואבסטרקציה, בעזרת ציור עצים פרקטלים באמצעות תווי ASCII. התוכנית שנחבר תפעל כך:

1. התוכנית תקרא ותעבד קובץ הכולל הצהרה, או הצהרות, של עצים פרקטלים. תיאור העצים הפרקטלים בהם עליכם לתמוך מוצגת בפרק 4.
2. התוכנית תדפיס את הפרקטלים שבקובץ ההצהרה שנקלט דרך ה־cli, אבל **בסדר הפוך** מסדר הופעתם בקובץ.

את פונקציית ה־main של התוכנית עליכם ליצור בקובץ FractalDrawer.cpp (ושם זה גם יהיה שם קובץ ה־executable).

3.1 קלט

התוכנית תקבל ארגומנט יחיד דרך ה־cli והוא נתיב לקובץ בפורמט CSV (Comma-separated values)[1]. בקובץ, כל שורה תתאר עץ פרקטלי שעלינו לעבד ולהדפיס. באשר לעמודות, אלו יהוא כדלקמן (הסדר משמאל לימין):

תא 1	תא 0
ממד הפרקטל	סוג העץ הפרקטלי

הערות והנחיות באשר לקובץ הפרקטלים:

- פורמט הקובץ הוא CSV, ניתן להניח כי הסיומת של הקובץ שתקבלו (במידה והוא תקין) היא csv. למשל (input.csv). קובץ עם סיומת שונה ייחשב כקלט לא תקין.
- **לא ניתן** להניח שהקובץ קיים, עליכם לוודא זאת.
- **לא ניתן** להניח שהקובץ אינו ריק, במקרה שהקובץ ריק - לא יודפס דבר.
- **ניתן** להניח שמדובר בקובץ CSV תקין. **מנגד, לא ניתן** להניח שיהיו רק שתי עמודות - עליכם לוודא זאת. קובץ שאינו עומד בפורמט המתואר **נחשב קובץ פגום** ועליכם לפעול בהתאם להוראות שיוצגו בהמשך (בפרק של טיפול בשגיאות).
- **לא ניתן** להניח שערכי העמודות תקינים. אם אחד מהערכים אינו תקין - מדובר בקובץ פגום. בפרט:

[1] ניתן להכיר את הפורמט ולראות דוגמה כאן: https://en.wikipedia.org/wiki/Comma-separated_values

- אם התבקש עץ פרקטלי שלא קיים בהתאם לפרק 4 לתרגיל, מדובר בקובץ פגום.
- אם ממד העץ קטן ממש מ־1, מדובר בשגיאה (כלומר, בהינתן ש־n הוא ממד העץ, אזי מתחייב ש־$n > 0$).
- אם ממד העץ גדול ממש מ־6, מדובר בשגיאה (כלומר, בהינתן ש־n הוא ממד העץ, מתחייב ש־$0 < n \leq 6$).

- הנכם **רשאים** להשתמש בספרית filesystem[2] של boost[3] וכן במחלקה tokenizer[4] של boost (אך אין להשתמש בספריות אחרות של boost). הרחבה בנוגע לשימוש ב־boost, לרבות הוראות התקנה ותוכנית לדוגמה, מופיעה בהמשך התרגיל.
- הנכם **חייבים** לעשות שימוש ב־STL (C++ Standard Template Library), ורשאים להשתמש בספריית cmath (זכרו לכלול בשעת ההידור את הדגל `-lm`).

3.2 ציור הפרקטלים

לאחר קריאת הקובץ, עליכם לצייר את הפרקטלים שהתבקשו בקובץ **בסדר הפוך לסדר הופעתם**. כפי שנוכחתם לראות מהגדרת קובץ ה־CSV, על המשתמש לבחור מבין מספר פרקטלים אפשרים - כולם מתוארים בפרק 4 לתרגיל. את הפרקטלים תציירו בעזרת ASCII Art (פלט של תווים למסך), כפי שיוסבר בהמשך. מכאן שאנחנו עובדים עם std::string ו־* char.
שימו לב לדגשים וההנחיות הבאות:

- התרגיל מכיל 3 קבצים שעליכם להגיש:
  - כל המחלקות הנוגעות לציור פרקטלים יישמרו ב־Fractal.h, Fractal.cpp.
  - ב C++ אנו יכולים לממש מספר מחלקות באותו קובץ. סגנון זה באופן כללי לא מומלץ אך עקב אילוצי המערכת אנחנו נעשה זאת בתרגיל הנוכחי.
  - הרצת התכנית והדפסת הפרקטלים למסך ימומשו בקובץ FractalDrawer.cpp.
- בתרגיל זה עיצוב התכנית שלכם מוגבל לאילוצים הבאים:
  - עליכם לשמור במחלקות הנוגעות ליצירת פרקטלים את לוח הפלט (ציור הפרקטל אותו לבסוף תדפיסו) **כוקטור** (`std::vector<T>`) כשדה של המחלקה. סוג האוביקטים (`T`) שאותם מחזיק הוקטור וכן הממד שלו (מימוש כוקטור או וקטור של וקטורים) נתונים להחלטתכם.
  - התוכנית **חייבת** להיות מעוצבת בצורה של תכנות מונחה עצמים.
- **שימו ♡**: עבדו לפי כללי **תכנות מונחה עצמים** שלמדתם בקורס זה ובייחוד בקורס מבוא לתכנות מונחה עצמים. חלק **משמעותי מאוד** מהניקוד שניתן לתרגיל הוא עבור עיצוב נכון וראוי, ושימוש בירושה.
- עיצוב התוכנה ייבדק באופן **ידני**. אי שימוש בעקרונות לעיל יגרור הורדת נקודות **משמעותית!**

[2] ראו: https://www.boost.org/doc/libs/1_70_0/libs/filesystem/doc/index.htm
[3] תוכלו לקרוא על הספריה המופלאה הזו כאן: https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries)
[4] ראו: https://www.boost.org/doc/libs/1_70_0/libs/tokenizer/doc/index.html

- חשבו האם אתם צריכים להשתמש בעקרונות כמו אינקפסולציה, פולימורפיזם, ואבסטרקציה? האם יש תבניות עיצוב (design patterns) שנכון להשתמש בהם בתרגיל זה? אם כן - אילו?

- **לעניין הפלט, שימו לב**: לאחר הדפסת כל פרקטל, **כולל** האחרון שמופיע בקובץ, יש להדפיס שורה אחת ריקה.

3.3 התמודדות עם שגיאות

עליכם לטפל במקרים בהם לא התקבל קלט תקין. אם מספר הארגומנטים שנשלחו לתוכנה אינו תקין, עליכם להדפיס ל־stderr את הפלט:

```
Usage:  FractalDrawer <file path>\n
```

מנגד, אם נתקלתם בקלט שגוי - דהיינו קובץ שאינו קיים או בקובץ פגום, עליכם להדפיס ל stderr:

```
Invalid input\n
```

בשני המקרים "\n" מסמן ירידת שורה (כפי שנהוג להדפיסה ב־C++). לאחר הדפסת הפלט, בשני המקרים עליכם לסגור את התוכנית (exit()) עם קוד סיום EXIT_FAILURE.

4 פרקטלים

להלן הפרקטלים בהם על תוכניתכם לתמוך. טרם ניגש לכך, שימו לב לדגשים הבאים:

- **מספר תת הסעיף יהיה מספר הפרקטל שעל המשתמש להזין.** לכן, למשל, Sierpinski Carpet מופיע בסעיף 4.1 ולכן כדי להדפיסו על המשתמש לבקש את מספר הפרקטל 1. לכן $fractal \in \{1, 2, 3\}$.

- נרצה להדפיס את הפרקטלים ב־ASCII Art, לכן נבחר את התווים בהם נשתמש:
  - כל פעם שנרצה לצייר חלק מהפרקטל, נשתמש **בתו סולמית** ("#").
  - כל פעם שנרצה לדלג ולא להדפיס, נשתמש **ברווח יחיד.**

- להלן תראו שנתאר כל עץ פרקטלי בצורה אינדוקטיבית, על $n \in \mathbb{N} \cup \{0\}$. עם זאת, בקובץ המתאר את רשימת העצים הפרקטלים שעל התוכנית להדפיס - לא נאפשר "לגשת" למקרה הבסיס (בו $n = 0$), ומקרים אלו יחשבו כקובץ פגום.

4.1 Sierpinski Carpet

זהו פרקטל שתואר לראשונה על ידי Wacław Sierpiński בשנת 1916. נתאר את בניית הפרקטל בצורה אינדוקטיבית על $n \in \mathbb{N} \cup \{0\}$:

- **בסיס**: כאשר $n = 0$, נתחיל בסימון יחיד (כלומר ריבוע אחד קטן, שאצלינו, כאמור, מסומן ב־#).

- **צעד:** בכל שלב ניצור 9 ריבועי משנה, שניתן לתאר בטבלה הבאה:

#	#	#
#		#
#	#	#

אם כך, במקרה הזה בכל שלב רקורסיבי, נוצר grid מסדר 3×3 **בו מציגים את תת־הפרקטל בסדר** $n-1$ **אם ורק אם הוא לא תת־הפרקטל האמצעי** (כלומר זה שבמיקום $(2, 2)$).
לדוגמה, בעוד כאשר הממד הוא 0 יודפס רק "#", כאשר הממד הוא 1, נקבל:

```
###
# #
###
```

וכאשר הממד הוא 2 נקבל:

```
#########
# ## ## #
#########
###   ###
# #   # #
###   ###
#########
# ## ## #
#########
```

4.2 Sierpinski Triangle

נתאר את בניית הפרקטל באופן אינדוקטיבי על $n \in \mathbb{N} \cup \{0\}$:

- **בסיס**: כאשר $n = 0$, נתחיל בסימון יחיד (כלומר ריבוע אחד קטן, שאצלינו, כאמור, מסומן ב־#).
- **צעד:** בכל שלב ניצור 4 ריבועי משנה, שניתן לתאר בטבלה הבאה:

#	#
#	

4.3 Vicsek fractal

פרקטל זה הוצע ע"י Tamas Vicsek וכיום משמש לייצור אנטנות קומפקטיות בין היתר.
נוכל לתארו באופן אינדוקטיבי על $n \in \mathbb{N} \cup \{0\}$ כך:

- **בסיס**: כאשר $n = 0$, נתחיל בסימון # יחיד.
- **צעד:** בכל שלב ניצור 9 ריבועי משנה, שניתן לתאר בטבלה הבאה:

#		#
	#	
#		#

5 חומר עזר - הספריה Boost

כאמור, בתרגיל זה הנכם רשאים לעשות שימוש בחבילות filesystem ו־tokenizer של boost.[5] Boost היא ספרית קוד־פתוח ב־C++ והמכילה חבילות (packages, או modules) רבות, מתחומים כדוגמת אינטרקציה עם קבצים, עבודה עם תכונות של מערכת ההפעלה, המקלות על הפיתוח. Boost היא ספריה פופולארית ביותר בתעשיה והמלצתינו החמה היא כי תקדישו מעט זמן להכירה - היכרות זו בהחלט תשתלם בעתיד!

5.1 התקנת הספריה

כדי לעבוד עם boost, עליכם ראשית להתקין את הספריה על מחשבכם האישי.

- להתקנת הספריה על מחשבים מבוססי Windows:
https://www.boost.org/doc/libs/1_70_0/more/getting_started/windows.html/.

- להתקנת הספריה במחשבים מבוססי Unix (כלומר Linux ו־macOS):
https://www.boost.org/doc/libs/1_70_0/more/getting_started/unix-variants.html.

שימו ♡: הספריה זמינה במחשבי בית הספר.

5.2 שימוש בחבילות מ־boost

באתר של boost תוכלו למצוא תיאור מלא, לרבות חותמות ה־API וכן **דוגמאות קוד** לשימוש בחבילות שונות Boost. בין היתר, תוכלו למצוא באתר גם דוגמאות קוד ותיאור ה־API של חבילת ה־filesystem וה־tokenizer, בהם הנכם רשאים להיעזר בתרגיל זה.
נציין כי בפרט, זכרו שכדי להשתמש ב־boost, עליכם לייבא את החבילה בה תרצו להשתמשת באמצעות פקודת `#include`. כך לדוגמה, כדי לעבוד עם tokenizer, יש לייבא את הספריה על ידי: `#include <boost/tokenizer.hpp>`.

5.3 קומפילציה

לאחר חיבור התוכנית, יהיה עליכם לוודא כי הקוד שלכם מהודר (מקומפל) כנגד החבילה שבה הנכם מעוניינים לעשות שימוש. כפי שלמדנו בחלק הראשון של הקורס (שעסק ב־C), עלינו לוודא כי המהדר "יודע לאתר" את כל קבצי ה־h בהם אנו נעזרים, וכמו כן יודע להדר את התוכנית כנגד הספריות הרלבנטיות (כקבצים בינארים):

- כדי לוודא שיש למהדר גישה לקבצי ה־h שעליהם נשענת הספריה, נשתמש בעת הקומפילציה בדגל `-L` ומיד לאחריו נציין את הנתיב לתיקיה שמכילה את הקבצים.

- כדי להגדיר כי יש להדר את תוכניתנו כנגד הספריה עצמה (דהיינו, הקובץ הבינארי המכיל את המימוש של הספריה), נשתמש בדגל `-l` ולאחריו נציין את שם הקובץ הבינארי.

לדוגמה, כדי להדר את הקובץ main.cpp כנגד החבילות filesystem ו־tokenizer, נוכל להשתמש בפקודה הבאה:

```
g++ main.cpp -o program -lboost_system -lboost_filesystem -L/path/to/boost
```

[5] לאתר הרשמי של boost: https://boost.org

לשימושכם, באתר הקורס ישנו קובץ Makefile לדוגמה וכן קובץ CMakeLists.txt לדוגמה - שניהם מאפשרים לעשות שימוש ב־boost.

לסיום, הערה באשר לקבצים עם הסיומת "hpp.": חדי העין יבחינו שישנם קבצים ב־boost שסיומם הוא "hpp.". מדובר בקבצים שמכילים - בנוסף להצהרות - גם מימוש.[6] מאחר שמדובר בוריאציה של קבצי h, אלו קבצים שלא עוברים הליך הידור עצמאי, ולכן לא יוצרים, כשלעצמם, קובץ בינארי. מנגד, קבצים אלו מכילים מימוש מלא, לכן נוכל לייבאם (בעזרת פקודת #include) ולעשות בהם שימוש מתוכניתינו (שהיא, כמובן, כן עוברת הליך הידור עצמאי ומלא). דוגמה לכך היא למשל הספריה tokenizer: הקובץ אותו יש לייבא בעניינה הוא tokenizer.hpp וכדי לעשות בה שימוש לא נדרש להדר את התוכנית כנגד ספריה כלשהי.

5.4 דוגמת שימוש

כדי להמחיש עד כמה פשוטה ועוצמתית boost, ויכולה לשרתכם נהימנה בתרגיל, בחרנו להביא דוגמת קוד אחת לשימוש בספריה.[7] לפניכם תוכנית C++ קצרה, העושה שימוש בחבילה tokenizer. בתוכנית, אנו מגדירים את המחרוזת "Boost C++ Libraries". לאחר מכן, באמצעות שימוש ב־tokenizer מפרידים את המחרוזת ל־tokens לפי התו רווח, כך שמתקבלת הקבוצה $\{"Boost",\ "C++",\ "Libraries"\}$. לבסוף, כל איבר בקבוצה מודפס בשורה נפרדת:

```
#include <boost/tokenizer.hpp>
#include <string>
#include <iostream>
int main() {
    typedef boost::tokenizer<boost::char_separator<char>> tokenizer;
    std::string s = "Boost C++ Libraries";
    boost::char_separator<char> sep{" "};
    tokenizer tok{s, sep};
    for (const auto &t:  tok) {
        std::cout << t << '\n';
    }
    return 0;
}
```

6 דוגמה

מאחר שמדובר בהדפסת ASCII לא נכלול בתיאור התרגיל דוגמה מורכבת. מומלץ **מאוד** לצפות בפתרון בית הספר מפני שפלט התוכנית לעתים **שונה** מתמונות הפרקטלים באינטרנט. עם זאת, בקצרה, אם נניח ש־f.txt מכיל את התוכן:

```
1,1
1,2
```

כלומר f.txt מצהיר על שני פרקטלים - Sierpinski Carpet מסדר 1 ומסדר 2. נקבל:

[6] נכלול את המימוש בקבצי "h", למשל, כאשר נעשה שימוש ב־templates, שאת מימושם **לא ניתן** לכלול בקבצי ה־cpp.

[7] הדוגמה נלקחה מהקישור הבא: https://theboostcpplibraries.com/boost.tokenizer.

```
$ FractalDrawer f.txt
#########
# ## ## #
#########
###   ###
# #   # #
###   ###
#########
# ## ## #
#########

###
# #
###
```

אשר שורה שנפתחת ב־$ מסמנת את הפקודה שהוקלדה. שימו לב לשורה הריקה בין הפרקטלים ובסוף הפלט.

7 נהלי הגשה

- קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס.
- **שימו לב:** עיצוב תוכנה - הבדיקה הידנית תהא **רחבת** היקף, אנא הקפידו לעצב את התוכנית כהלכה ובהתאם לעקרונות שנלמדו. ציון תרגילים שלא יממשו עקרונות OOP כהלכה יפגע **באופן משמעותי מאוד**, גם אם הם עומדים במלוא הבדיקות האוטומטיות.
- **שימו לב:** מאחר שתוכניתכם מדפיסה ערכי ASCII ונבדקת באופן אוטומטי, עליכם לדייק בהדפסה. זכרו להשתמש אך ורק בתו "#" להדפסת הפרקטלים, ובתו רווח (יחיד) לתא ריק ב־grid. בסופו של דבר, עבור כל פרקטל, תצטרכו להדפיס כמות תווים הזהה לכמות התווים ב־grid. לכן, למשל, עבור Sierpinski Carpet מממד 1, הכולל grid של 3×3, נקבל הדפסה של 9 תווים ועוד 3 ירידות שורה (אחת בסוף כל שורה).
- זכרו שעליכם לקמפל את התוכנית כנגד מהדר לשפת C++ בתקן שנקבע בקורס. כמו כן, זכרו שעליכם **להעדף** פונקציות ותכונות של C++ על פני אלו של C. למשל, נעדיף להשתמש ב־`new` ו־`delete` על פני `malloc` ו־`free`, וכן נעדיף להשתמש ב־`std::string` מאשר ב־`char*`.
- **נזכיר:** כאמור בהנחיות הכלליות להגשת תרגילים ־ הקצאת זיכרון דינמית מחייבת את שחרור הזיכרון, למעט במקרים בהם ישנה שגיאה המחייבת סגירת התוכנית באופן מיידי עם קוד שגיאה (כלומר קוד יציאה השונה מ־0). תוכלו להיעזר בתוכנה valgrind כדי לחפש דליפות זיכרון בתוכנית שכתבתם.
- פתרון בית הספר זמין בנתיב

  `~labcc/www/cpp_ex2/FractalDrawer`

- בתרגיל זה אין להגיש קובץ Makefile. עליכם ליצור קובץ tar הכולל **אך ורק** את הקבצים FractalDrawer.cpp, Fractal.h, Fractal.cpp. ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
$ tar -cvf cpp_ex2.tar FractalDrawer.cpp Fractal.cpp
Fractal.h
```

שימו לב: קבצי קוד המקור שתכתבו נדרשים להתקמפל כהלכה עם std++14, כנדרש בהוראות להגשת תרגילים שפורסמו באתר הקורס.
אם בחרתם להשתמש ב־boost, עליכם לוודא שהגרסה שעמה עבדתם נתמכת במחשבי בית הספר.

- אנא וודאו כי התרגיל שלכם עובר את ה־Pre-submission Script **ללא שגיאות או אזהרות**. קובץ ה־Pre-submission Script זמין בנתיב.

```
~labcc/www/cpp_ex2/presubmit_cpp_ex2
```

- על תרגילכם לעמוד בדרישות ה־Coding Style של הקורס. פידבק בנושא זה תקבלו כחלק מה־Pre Submission. נוסף לכך, תוכלו לבדוק את תוכניתכם גם באמצעות הרצת הסקריפט בנתיב:

```
~labcc/www/codingStyle
```

בהצלחה!!