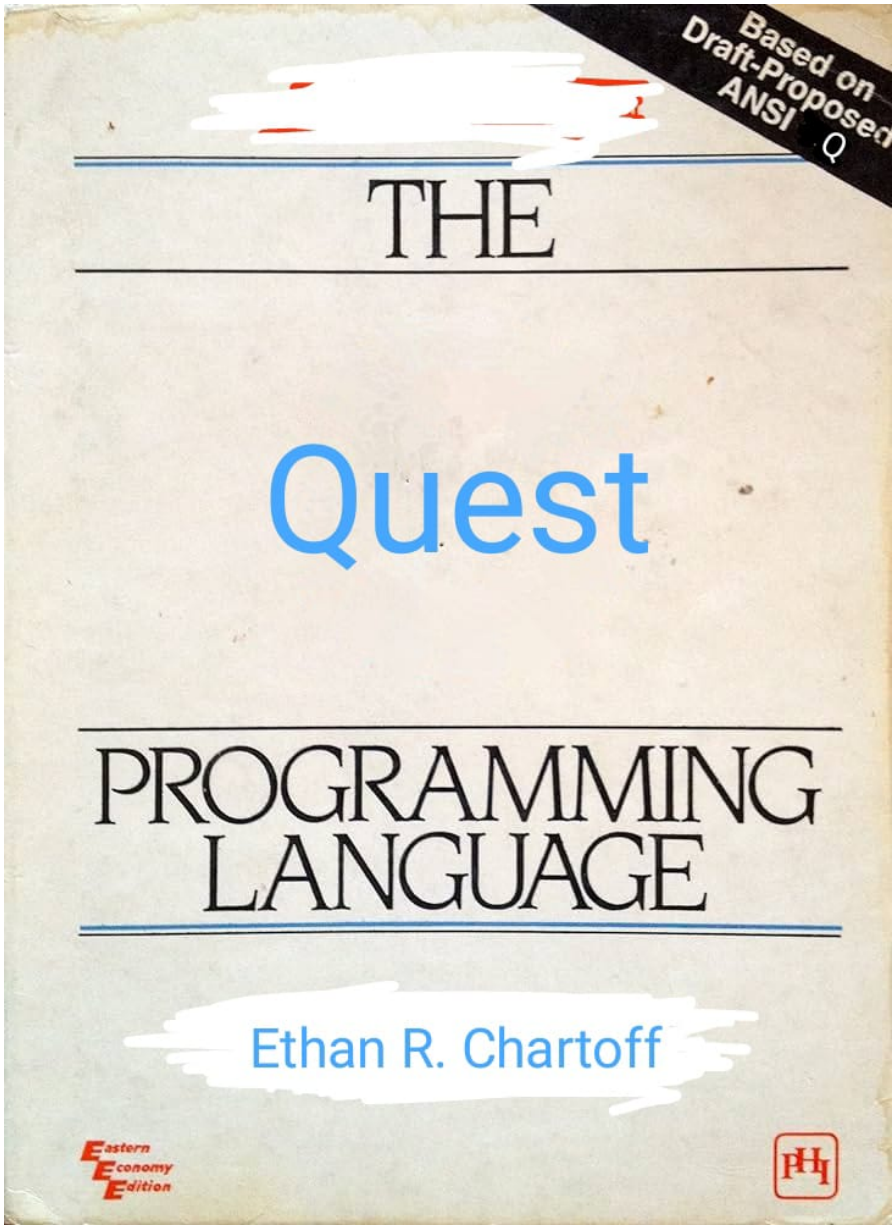
הצעת פרויקט - י"ג הנדסת תוכנה



סמל מוסד: 471029
שם המכללה: מכללת אורט הרמלין נתניה
שם הסטודנט: איתן רפאל צ'רטוף
ת"ז הסטודנט: 215310715
שם הפרויקט: The Quest Programming Language

רקע תיאורטי בתחום הפרויקט

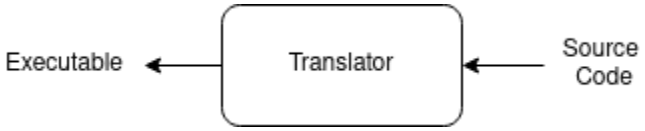
שפת תכנות

שפת תכנות היא קבוצה של סימונים המשמשת לכתיבת תוכניות מחשב. כל סימון מפיק תוצאה, וביחד הסימונים יכולים לממש אלגוריתמים שונים. תוכניות המחשב הם קבוצה של הוראות שהמחשב יכול לבצע. יש הרבה שפות תכנות שונות, כל אחת עם דקדוק וסמנטיקה משלהם (המוגדרים ע"י שפה פורמלית במתמטיקה). שפות תכנות הם חלק בלתי נפרד ממדעי המחשב, עם סניף משלהם הנקרא "תורת שפת התכנות".

שפות תכנות משמשות אותנו כמעט בכל מקום, לדוגמה שירות הסטרימינג Netflix משתמש בשפות תכנות רבות כמו Java וPython בשביל להביא להביא ללקוחות שלו וידאו, Google המשתמשים בC++ לרוב הפרויקטים הפתוחים שלהם, ואפילו פיתחו שפות תכנות כמו Go.

יש להבדיל בין שפות תכנות לשפות מדוברות כמו עברית, מכיוון ששפות תכנות נועדו לתקשר הוראות למחשב בשפה שבני אדם יוכלו להבין, ושפות מדוברות נועדו לתקשורת בין אנשים. למרות זאת לשני סוגי השפות כן יש דברים דומים, לדוגמא תחביר ודקדוק. יש להבדיל גם כן בין שפות מחשב ושפות תכנות. שפת תכנות היא סוג של שפה המשומשת במחשוב, ושפת מחשב יכולה להתייחס לכל שפה המשומשת לתקשור עם מחשב.

בשביל שהמחשב יוכל להבין את סימוני השפה ולממש את תכנית המחשב, הוא צריך מערכת המתרגמת את סימוני השפה להוראות אשר המחשב יכול לבצע. מערכות אלו נקראות מתרגמי/מעבדי שפת תכנות.



צריך לדעת שלא לכל המחשבים יש את אותו קבוצת הוראות הביצוע, ומה שמחשב יכול לבצע תלוי בארכיטקטורה שלו.

סוגי מעבדי שפות תכנות

מתורגמן/Interpreter

האינטרפרטר הינה תכנית מחשב המבצעת ישירות הוראות שנכתבו בשפות תכנות מבלי לדרוש שהן תורגמו לשפת מכונה. בדרך כלל האינטרפרטר כולל קבוצה של הוראות שאפשר לבצע ורשימה של הוראות אלו לפי הסדר שהתכניתן רצה שההוראות יפעלו.
האינטרפרטר מתרגם ומבצע את התכנית הרצויה שורה אחרי שורה, לכן בדרך כלל האינטרפרטרים יהיו איטיים מקומפיילרים, המתרגמים את כל התכנית.

מהדר/Compiler

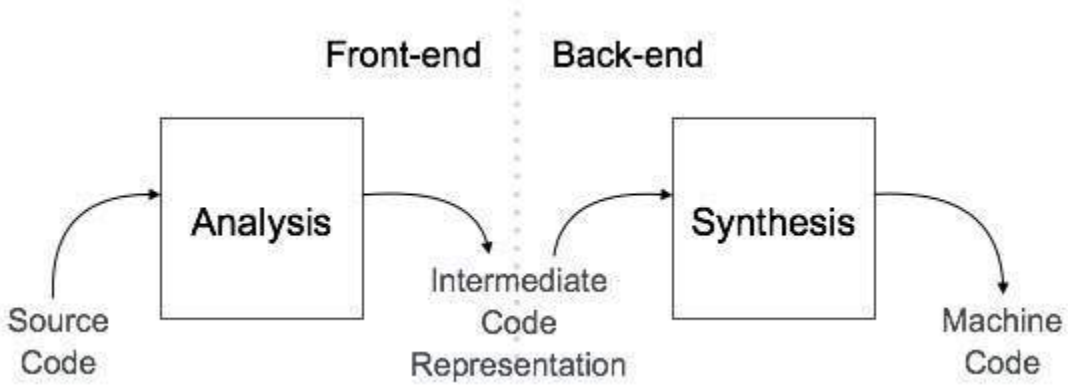
הקומפיילר הינה תכנית מחשב המתרגמת קוד[1] משפה אחת לשפה אחרת. בדרך כלל משתמשים בקומפיילרים בשביל לתרגם קוד משפת תכנות ברמה גבוהה לשפת תכנות ברמה נמוכה[2]. רוב הקומפיילרים יממשו את התהליכים הבאים:

- עיבוד מקדים (preprocessing): תכנית הקולטת נתונים מקדימים בשביל שהפלט שלה ישמש בתכנית אחרת. סוג תכנית זו תקרא תמיד לפני תכנית אחרת שתשתמש בפלו תכנית זאת, לכן השם עיבוד מקדים. דוגמה פשוטה לסוג תכנית זו אפשר למצוא בקומפיילר של C, שחלק אחד שה-preprocessor עושה הוא לקחת את כל השורות המתחילות עם '#' והופך אותם להוראות בשביל הקומפיילר.

- ניתוח מילוני (lexical analysis): בחלק זה התכנית קוראת את הקוד והופכת אותו אל אסימונים או "lexical tokens", שהם כמו היחידות הבסיסיות של השפה. אפשר לקרוא לתכנית זו ה"lexer".

- ניתוח תחביר (syntax analysis/parsing): חלק זה קולט את האסימונים שקיבלנו מהתכנית הקודמת ומחיל אליהם כללים מוגדרים כדי לקבוע עם הקוד נכון מבחינה תחבירית. קוראים לתכנית זו ה"parser".

- ניתוח סמנטי (semantic analysis): בשלב זה התכנית בוחנת האם לקוד יש שגיאות סמנטיות כמו טיפוסים לא תואמים או משתנה שלא הוגדר.



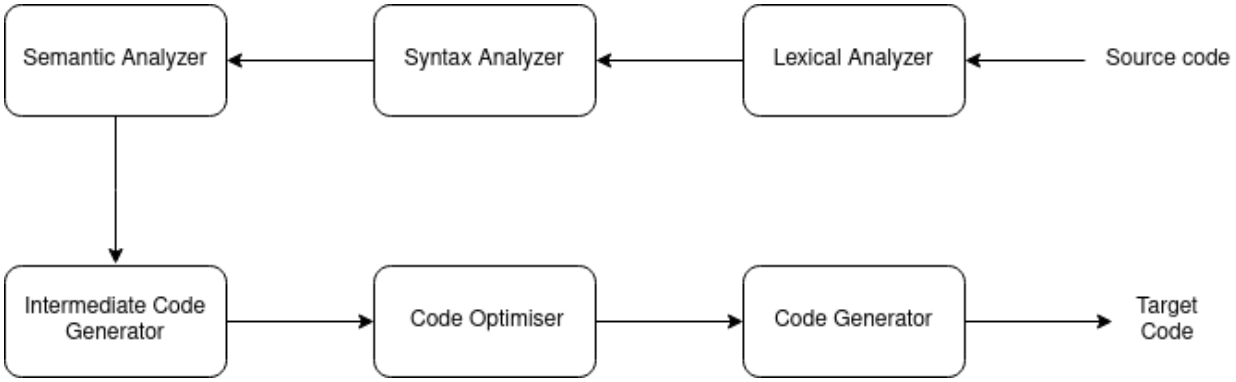
- יצירת קוד ביניים (intermediate code generation): לאחר ניתוח הקוד, התכנית מייצרת פרזנטציה שונה של הקוד ופולטת אותה בשביל שהקומפיילר אשתמש בא לאופטימיזציה ותרגום.

- אופטימיזציה (code optimization): שלב זה משתמש בפלט של התכנית הקודמת ומשפר אותה. תכנית זו עושה דברים כמו מוחקת שורות קוד מיותרות ומסדרת את ההוראות בשביל למהר התכנית הסופית.

- יצירת קוד (code generation): בשלב זה התכנית קולטת את הקוד המשופר ומתרגמת אותו לשפת המחשב הרצויה.

---

[1] קוד - קבוצה של סימנים המשמשים כהוראות התכנית.
[2] שפות תכנות ברמה גבוה ונמוכה - שפות תכנות ברמה גבוה הם שפות תכנות עם הרבה הפשטה מן פרטי המחשב, בשביל שהיו קריאות יותר לאנשים. שפות תכנות ברמה נמוכה הם שפות עם קצת הפשטה מן פרטי המחשב, והוראה אחת מתורגמת ישירות להוראת שפת מכונה.



מושגי תכנות בסיסיים

משתנים

מקום אחסון בעל שם וסוג ערך שמור. אפשר להתייחס למשתנים בעזרת שמם או כתובת הזיכרון שלהם. בזמן ריצת תכנית מחשב אפשר להכין משתנים, להגדיר להם ערך, לשנות ערך זה, למחוק את המשתנים ועוד.
דוגמאות להגדרת משתנים שישמשו כמידע על בן אדם:

```
1. // גיל המוגדר כמספר שלם
2. int age;
3. // שם המוגדר כמחרוזת של אותיות
4. string name;
5. // מספר אהוב המוגדר כשבר
6. float fav_number;
```

תנאים

הוראה הבודקת תנאי מסוים. תנאים הם דרך לבדוק תנאי מסוים בזמן ריצת התכנית, ואפשר להשתמש במשתנים בתנאים. תנאים בדר"כ כתובים בהוראות if - else, הכוונה היא אם קורה משהו, תעשה משהו, ואם לא תעשה משהו אחר:

```
1.  if (תנאי):
    1.1.  (הוראות)
2.  else:
    2.1.  (הוראות שונות)
```

לולאות

לולאות משומשות בשביל להריץ חלק של הקוד שוב ושוב עד שתנאי מסוים מתקיים. יש שני סוגים עיקריים של לולאות, ה - for loop וה - while loop. בדר"כ משתמשים בfor שאנחנו יודעים את מספר האיטרציות, ובwhile שאנחנו לא. דוגמא לשני תוכניות המדפיסות 10 כוכביות, אחת לאחר השנייה:

```
1.    int i;
2.    for(i = 0; i < 10; ++i) {
      2.1.    printf("*");
3.    }
```

```
1.    int i = 0;
2.    while(i < 10) {
      2.1.    printf("*");
      2.2.    ++i;
3.    }
```

פונקציות

פונקציות הינם כמה הוראות המבצעות משימה מסוימת, שאפשר לקרוא להם בקריאה אחת כאשר אנחנו רוצים לבצע את אותה המשימה שהפונקציה מבצעת. פונקציות יכולות לקבל משתנים ולהחזיר ערכים.
דוגמא לפונקציה המקבלת שני שלמים ומחזירה את הסכום שלהם:

```
1.    int sum(int a, int b) {
      1.1.    return a + b;
2.    }
```

- סוגריים מסולסלים מגדיר את היקף הקוד, כלומר איזה הוראות נעשו אחרי הוראה ספציפית ואיפה שאפשר להתייחס למשתנה. ערכים הם נגישים רק בתוך ההיקף שהם מוגדרים בוא.
- טקסט לאחר הסימון // או שורות בתוך הסימונים /* */ מגדירים הערות, טקסט שאפשר לשים ליד קוד בשביל להסביר את התכנית בשפה שבני אדם מבינים.

```c
/*
 * This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world\n" to
 * standard output when invoked.
 */
void sayHello(void) {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!\n");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
int main(void)
{
    // Invoke the sayHello() function.
    sayHello();
    return 0;
}
```

תכנית הכתובה בשפת התכנות C. כאשר נהדר את התכנית ונריץ אותה, התכנית תפלוט "Hello, world!".

פרדיגמות (שפות) תכנות

פרדיגמת תכנות היא דרך חשיבה על משימת כתיבת תוכנות מחשב וגישה אליהן. פרדיגמות תכנות כוללות מערך של עקרונות, שיטות ופרקטיקות המנחים עיצוב תוכנה. פרדיגמות תכנות שונות התפתחו עם הזמן כדי לתת מענה לאתגרים שונים בפיתוח תוכנה, ולכל אחת מהן קבוצת מושגים וטכניקות משלה. רוב שפות התכנות כיום יתנו לתוכניתן להשתמש ביותר מפרדיגמה אחת, אך בנויות עם הכוונה שישתמשו רק באחת מן מהן.

תכנות פרוצדורלי

תכנות פרוצדורלי הינו פרדיגמת תכנות המבוססת על קריאות לפרוצדורות (פונקציות). המושגים העיקריים של תכנות פונקציונלי הם:

- פרוצדורות: פרוצדורות יכולות להיקרא מכל מקום בתוכנית (אם הגדירו אותם פרוצדורות), כולל מתוך הפרוצדורה עצמה, מה שיגרום לרקורסיה[3].
- הפרדה בין ערכים ופונקציות: משתנים ופרוצדורות מופרדים אחד מהשני, אך מידע יכול להיכנס אל פרוצדורה עם הארגומנטים הנכונים, ופרוצדורות יכולות להחזיר ערכים.
- היקפיות: מגדיר איפה משתנים מסוימים יכולים להיות משומשים.

תכנות מונחה עצמים

תכנות מונחה עצמים הינו פרדיגמת תכנות המבוססת על עצמים (אובייקטים). אובייקטים מאחסנים אוספים של מידע וקוד שבדר"כ מבצע פעולות על אוספי המידע. הסוג הכי פופולרי של תכנות מונחה עצמים הוא תכנות עצמים בעזרת מחלקות, שבעזרתם אפשר להגדיר משתנים ופרוצדורות. בתכנות עצמים בעזרת מחלקות אובייקטים הם מופעים של מחלקות, ומכיוון שאובייקטים הם משתנים, המחלקות הם סוגם. המושגים העיקריים של תכנות מונחה עצמים הם:

- עצמים: מופעים של מחלקות. בכל מופע יכול להיות אוסף מידע שונה ופרוצדורות שונות. כל עצם הינו ספציפי לעצמו.
- מחלקות: סוגי נתונים המוגדרים על ידי התכניתן הפועלים כמתווה עבור עצמים, פרוצדורות ומשתנים.
- כימוס (Encapsulation): חיבור של משתנים ופרוצדורות השייכות למחלקה. רעיון זה מגביל גישה ישירה לחלק ממרכיבי האובייקט ויכול למנוע שינוי מקרי ולא רצוי של נתונים.
- הורשה (Inheritance): הורשה של מאפיינים ממחלקה קיימת למחלקה חדשה. הורשה מאפשרת לתוכניתן ליצור היררכיה של מחלקות, המעתיקות ומוסיפות ממחלקת ההורה שלהם.
- רב צורתיות (Polymorphism): פולימורפיזם מאפשר להתייחס לאובייקטים ממחלקות שונות כאובייקטים של מחלקת בסיס משותפת. בדר"כ נעשה באמצעות דריסת פרוצדורות וממשקים שהורשו.

תכנות פונקציונלי

תכנות פונקציונלי הינו פרדיגמת תכנות המתייחסת למחשוב כהערכת פונקציות מתמטיות. המושגים העיקריים של תכנות פונקציונלי הם:

- פונקציות מסדר גבוהה: פונקציות שיכולות לקבל פונקציות אחרות כפרמטרים ו\או להחזיר אותם כתוצאות.
- פונקציות מדרגה ראשונה: פונקציות היכולות להופיע בכל מקום בתכנית שמשתנים יכולים, ויתייחסו אליהם כמו משתנים.

---

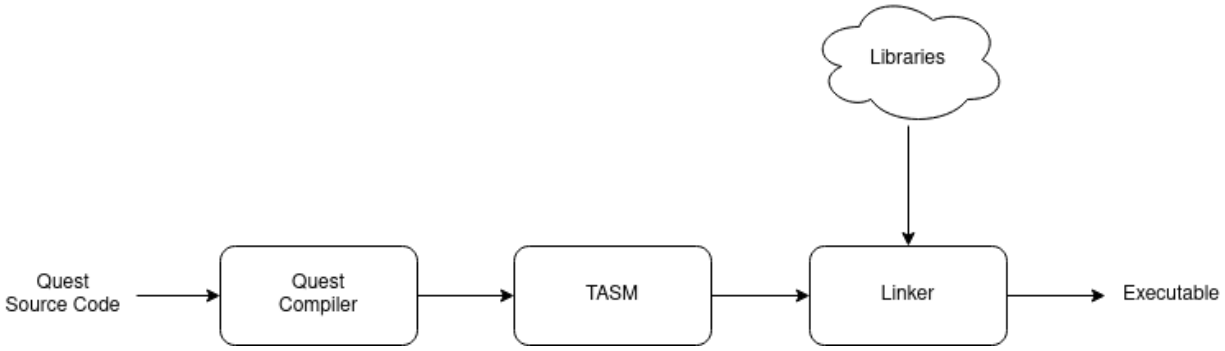
[3] רקורסיה - במדעי המחשב, רקורסיה הינה טכניקה שבא פונקציות קוראות לעצמם.

- פונקציות טהורות: פונקציות המחזירות את אותן התוצאות עבור אותם המשתנים כל פעם, והפונקציה לא משנה ערכים לא לוקלים (מקומיים) או סטטים[4].
- רקורסיה: איטרציה נעשית ע"י רקורסיה, בגלל שאין לולאות בתכנות פונקציונלי.
- הערכה עצלנית: ההערכה של ביטויים נעשים אך ורק מתי שהערך נחוץ.
- משתנים לא ניתנים לשינוי עם סוג: משתנים בשפות תכנות פונקציונליות משתנים אינם ניתנים לשינוי, כלומר, לאחר שהגדרנו ערך למשתנה מסוים, אי אפשר לשנות ערך זה. בנוסף, למשתנים יש סוג, כמו שלמים, חרוזים, מערכים ועוד...

יש לשים לב שפונקציות בפרדיגמת תכנות זו הם שונות מפונקציות מפרדיגמות אחרות.

תיאור הפרויקט

הפרויקט שלי יהיה מהדר שאני אצור מכלים פשוטים. המהדר מתרגם שפה מומצאת שאני אצור הנקראת "Quest" אל שפת אסמבלי. משמשים יוכלו להשתמש בשפה בשביל ליצור תכניות מחשב, וכאשר הם רוצים לתרגם את התכנית בשביל שהמחשב יבצע אותה, הם ישתמשו במהדר המתואר.
המהדר יתרגם את השפה המומצאת אל שפת אסמבלי, ולאחר מכן אשתמש באסמבלר tasm בשביל להעביר לקובץ הרצה.



הגדרת השפה

השפה תהיה שפת תכנות פרוצדורלית, יהיו בא הוראות סטנדרטיות כמו:

- משתנים - שלמים, תווים וכו...
- פונקציות - עם סוגי החזרה ופרמטרים לקבלה
- תנאים - if - else, switch...
- לולאות - for, while...
- מערכים
- I/O
- מצביעים
- ועוד...

[4] משתנים לוקליים וסטאטים - משתנים לוקליים הינם משתנים המוגדרים בהיקף מסוים. לאחר שהתכנית יוצאת מהיקף, כל המשתנים שאולקץ להם זיכרון נמחקים, אלה אם המשתנה סטטי.

משתמשים בשפה יכלו לבנות אלגוריתמים כמו האלגוריתמים מהסעיפים הקודמים בעזרת כל סוגי ההוראות הקודמים. השפה תיחשב לשפה "Turing complete", כלומר שפת תכנות בעלת יכולת לדמות מכונת טיורינג[5]

הגדרת הבעיה האלגוריתמית

הבעיה האלגוריתמית של הפרויקט היא איך המהדר אתרגם את השפה המומצאת אל שפת מכונה. כלומר, הבעיה מחולקת לשבעה החלקים המוגדרים בשלבי המהדר, ולכל שלב כזה ישנם עוד בעיות קטנות משל עצמו, לדוגמה, איך הlexer יכין את האסימונים, איך הparser אבחון את כל האסימונים וכו...

בעיות אלגוריתמיות:

- Preprocessing: מימוש נתונים מקדימים והעברת פלט שהחלקים האחרים יוכלו להיעזר בוא.
- Lexing: תרגום הקוד לאסימונים.
- Parsing: מימוש כללי השפה, בדיקת נכונות הקוד.
- Semantic Analysis: בדיקת שגיאות סמנטיות.
- Intermediate Code Generation: מימוש הפרזנטציה השונה של הקוד ותרגומו.
- Optimization: מימוש כללי אופטימיזציה, מציאת קטעי קוד שאפשר לשפר אותם.
- Generation: תרגום לשפת מכונה.

תהליכים עיקריים בפרויקט

1. חקר מקדמים על סוגים שונים של מהדרים והבנייה שלהם.
2. הגדרת השפה הפורמלית.
3. בניית כל שלב המהדר.
4. תיקון שגיאות, אופטימיזציה של האלגוריתם ובחינתה.

שפות תכנות

המהדר יכתב בשפת התכנות C, בנוסף לספריות עזר בסיסיות של השפה.

סביבות עבודה

סביבות העבודה אשר המהדר יכתב בהם הם:

- סביבת המתכנת האינטראקטיבית VScode,
- עורך הטקסט NeoVim.

לוח זמנים

12/11/23 - הגשת נוסח ראשוני (first draft) - הצעת ותיאור הפרויקט.
18/11/23 - הגשת נוסח סופי (final draft) - הצעת ותיאור הפרויקט.

[5] מכונת טיורינג - מודל חישוב מתמטי המתאר מכונה מופשטת המבצעת מניפולציות על סמלים על גבי רצועת סרט לפי טבלת כללים. למרות פשטות הדגם, מכונה זו יכולה ליישם כל אלגוריתם מחשב (אם נתון מספיק זמן ומשאבים).

01/12/23 - הגשה למשרד החינוך ואישור.
01/12/23 - מחקר על מהדרים ומימוש השפה.
ינואר - בניית מנתח מילוני מנתח תחבירי ומנתח סמנטי.
26/01/24 - דו"ח
פבואר - בניית מעבד מקדים ויוצר קוד ביניים.
23/02/24 - הגשת דו"ח ביניים - התקדמות עד כה.
מרץ - בניית יוצר קוד ומנתח לאופטימיזציה.
אפריל 24 - סיום פיתוח (code freeze).
אפריל 24 - בחינת מתכונת כולל תיק פרויקט.
אפריל 24 - סיום תיקון שגיאות (bug freeze) - סופי.
אפריל 24 - הגשת תיק מלווה של הפרויקט הסופי.
אפריל 2024 - הגנה על עבודת הגמר - פנימית - סופית.
מאי 24 - הגנה על עבודת הגמר - חיצונית.

חתימות

חתימת הסטודנט:

X _____

חתימת רכז המגמה:

X _____

אישור משרד החינוך:

X _____