Mazes

איתמר רייף

מדעי המחשב / Assembly 8086

גימנסיה הרצליה / שלומי אחנין

20 במרץ 2015

תוכן עניינים

מבוא, מטרות וקשיים בדרך

Mazes הוא משחק קלאסי שמטרתו לחמוק מהמחסומים ולהגיע לסוף המבוך. בסוף המבוך ייטען שלב חדש, או במקרה שזהו השלב האחרון – מסך הנצחון.

המשחק עצמו הוא קלאסי ולא מסובך, אך מראה את יישומם של עקרונות בסיסיים בשפת Assembly 8086 (כגון שימוש בעכבר, ניצול ה-video mode והדגמת השיטה שלפיה מחרוזת מאוחסנת בזכרון).

תהליך עשיית הפרויקט היה מעניין ומאתגר, והידע שרכשתי הוא עצום ובעל ערך רב מאוד עבורי. אני מאמין כי הצלחתי לבנות פרויקט המדגים בצורה יסודית ועמוקה את העקרונות שלמדתי ואת עמודי היסוד של השפה Assembly 8086.

הוראות המשחק

תיאור המשתנים

mazes	מערך המכיל את המחרוזות של השלבים במשחק
victory	מחרוזת של מסך הנצחון (משמש להדפסתו)
titleScreen	מחרוזת של מסך הפתיחה (משמש להדפסתו)
cheatScreen	מחרוזת של מסך ה-"cheats" (משמש להדפסתו)
changeStartScreen	מחרוזת של מסך שינוי שלב ההתחלה (משמש להדפסתו)
changeCharacterScreen	מחרוזת של מסך שינוי השחקן (משמש להדפסתו)
errorScreen	מחרוזת של מסך ה-"error" (משמש להדפסתו)
x	מספר הטור של ה-cursor
y	מספר השורה של ה-cursor
xCheck	מספר הטור אליו ה-cursor צריך לעבור – עובר לבדיקת המיקום החדש
yCheck	מספר השורה אליו ה-cursor צריך לעבור – עובר לבדיקת המיקום החדש
posInfo	הערך שיש במיקום ה-cursor
posInput	הערך של המקש שהוקש לשינוי מיקום ה-cursor
mouseX	ה-X של מיקום העכבר AX=03h/INT 33h
mouseY	ה-Y של מיקום העכבר AX=03h/INT 33h
mouseClick	מצב העכבר בעת ביצוע AX=03h/INT 33h – איזה כפתור נלחץ
startMPos	מסמן האם העכבר נלחץ על כפתור התחל במסך הפתיחה
checkCheatStart	מסמן אם העכבר נלחץ על כפתור שינוי שלב במסך cheats
checkCheatCharacter	מסמן אם העכבר נלחץ על כפתור שינוי דמות במסך cheats
mazesIndex	מסמן את ה-index של השלב הנוכחי במערך mazes
mazeCount	מסמן באיזה שלב נמצא המשתמש
$Count	סופר כמה שלבים סיים המשתמש לשם השוואה עם ה-index
character	קוד ה-ASCII של דמות המשתמש

01h מיקום טוב

0h מיקום לא טוב

תיאור הפרוצדורות

פרוצדורות בסיסיות של התכנית

beginning	מדפיס את מסך ההתחלה של המשחק.
startMouse	מקבל קלט עכבר עבור מסך ההתחלה של המשחק.
cheats	מדפיס את מסך ה-cheats.
cheatsMouse	מקבל קלט עכבר עבור מסך ה-cheats של המשחק.
changeStart	כאשר כפתור שינוי שלב ההתחלה במסך ה-cheats נלחץ, הפרוצדורה מדפיסה את המסך עבור שינוי שלב ההתחלה, מקבלת קלט של מספר ומציבה אותו במשתנה mazeCount ואז קוראת לפעולה newLevel.
changeCharacter	כאשר כפתור שינוי שלב דמות השחקן ה-cheats נלחץ, הפרוצדורה מדפיסה את המסך עבור שינוי דמות השחקן, מקבלת קלט של מספר ומציבה אותו במשתנה character ואז קוראת לפעולה newLevel.
goodStartMPos	כאשר נלחץ כפתור ה-start במסך ההתחלה, הפרוצדורה מדפיסה את השלב הראשון ובאמצעות הפרוצדורה output מדפיסה את דמות השחקן.
posChange	מקבל קלט באמצעות invisInput ולפיו קורא לפרוצדורת הזזת השחקן המתאימה.
exit	מאפס את המסך ויוצא מהתכנית.

פרוצדורות אלו הן השלד של התכנית. הן אלו שקובעות את סדר הפעולות בתכנית ואליהן חוזרות הפרוצדורות שנקראו כאשר אלו מסיימות את תפקידן.

פרוצדורות I/O בסיסיות

invisInput	מקבל קלט בלי להשאיר אותו על המסך.
output	מדפיס את דמות שחקן על פי הערך character על המסך.
posChangeSW	מזיז את השחקן למטה ושמאלה.
posChangeS	מזיז את השחקן למטה.
posChangeSE	מזיז את השחקן למטה וימינה.
posChangeE	מזיז את השחקן ימינה.
posChangeW	מזיז את השחקן שמאלה.
posChangeNW	מזיז את השחקן למעלה ושמאלה.

posChangeN	מזיז את השחקן למעלה.
posChangeNE	מזיז את השחקן למעלה וימינה.

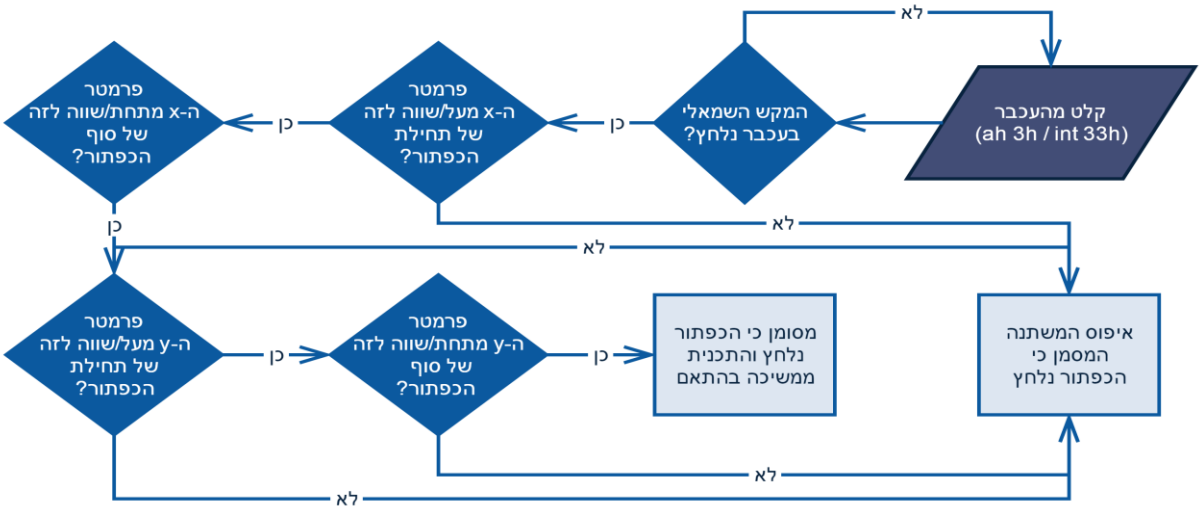
פרוצדורות אלו מבצעות קלט ופלט בסיסיות בתכנית (קלט של ערך, הדפסה של דמות השחקן, הזזת הדמות). הצורה שבה פועלת הפרוצדורות המזיזות את השחקן היא זהה: הפרוצדורה מתחילה באיפוס התא בו נמצא ה-cursor ואז משנה את המשתנים (xCheck, yCheck) המשמשים לבדיקת התא אליו השחקן רוצה לזוז בהתאם לקלט (אם השחקן רוצה לזוז למעלה אז מחסרים 1 מ-xCheck. אם השחקן רוצה לזוז ימינה ולמטה מעלים את xCheck ב-1 ואת yCheck ב-1 וכו'). לאחר מכן הפעולה checkPos נקראת. אם היעד אליו רוצה השחקן לעבור פנוי (פירוט תחת checkPos), הפרוצדורה משנה את המשתנים המחזיקים את הערך הנוכחי של המיקום (x, y) בהתאם ואז ה-cursor מוזז בהתאם. בסוף, הפעולה חוזרת ל-posChange.

פרוצדורות עכבר

checkStartMClick	בודק האם הכפתור הימני של העכבר נלחץ. אם כן, ממשיך לבדוק את מיקום הלחיצה. אם לא – חוזר ל-startMouse, הפרוצדורה מתחילה את תהליך בדיקת הלחיצה במסך הראשי.
checkMouseX	בודק האם פרמטר ה-x של הלחיצה גדול/שווה ל-36h (התחלת "הכפתור" השמאלי ביותר). אם כן, ממשיך לבדוק את מיקום הלחיצה. אם לא – חוזר ל-startMouse.
afterStartX	בודק האם פרמטר ה-x של הלחיצה קטן/שווה ל-0B8h (סוף "הכפתור" start). אם כן, ממשיך לפעולה goodStartXPos. אם לא – עובר לפעולה checkCheatX.
goodStartXPos	מסמן כי הפרמטרים של הלחיצה תואמים את התחום של "כפתור" ה-start עד כה וממשיך לאימות פרמטר ה-y של הלחיצה.
checkStartMouseY	בודק האם פרמטר ה-y של הלחיצה גדול/שווה ל-36h (התחלת "כפתור" ה-start). אם כן, ממשיך לבדוק את מיקום הלחיצה. אם לא – חוזר ל-startMouse.
afterStartY	בודק האם פרמטר ה-y של הלחיצה קטן/שווה ל-58h (סוף התחום של "כפתור" ה-start). אם כן, ממשיך לפרוצדורה goodStartYPos. אם לא, בגלל שכבר התקבל שפרמטר ה-x תואם גם את התחום של כפתור ה-cheat, ממשיך לפעולה checkCheatY1.
goodStartYPos	מסמן כי הפרמטרים של הלחיצה תואמים את מיקום "כפתור" ה-start וחוזר לקריאה המקורית (ret).
checkCheatX	בודק האם פרמטר ה-x של הלחיצה תואם את זה של כפתור ה-cheats. אם כן, ממשיך לפעולה cheats (הפעולה המתחילה את סדר הפעולות של מסך ה-cheats). אם לא – עובר לפעולה badStartMPos.
checkCheatY1	בודק האם פרמטר ה-y של הלחיצה תואם את זה של כפתור ה-cheats. אם כן, ממשיך לפעולה checkCheatsY2. אם לא – עובר לפעולה badStartMPos.
checkCheatY2	בודק האם פרמטר ה-y של הלחיצה תואם את זה של כפתור ה-cheats. אם כן, ממשיך לפעולה cheats. אם לא – עובר לפעולה badStartMPos.
badStartMPos	מסמן כי הלחיצה לא היתה בתחומים של כפתור כלשהו ועובר ל-startMouse.
checkCheatsMClick	בודק האם התבצעה לחיצה. אם כן – ממשיך לפעולה checkCheatsMouseX1. אם לא – עובר ל-cheatsMouse.

checkCheatsMouseX1	משווה את פרמטר ה-x של הלחיצה לתחילת הכפתור הראשון. אם הוא תואם – ממשיך ל-afterCheatsX1. אם לא – עובר ל-checkCheatsMouseX2.
afterCheatsX1	משווה את פרמטר ה-x של הלחיצה לסוף הכפתור הראשון. אם הוא תואם ממשיך ל-afterCheatsY11. אם לא, עובר ל-checkCheatsMouseX2 .
checkCheatsMouseX2	משווה את פרמטר ה-x של הלחיצה לתחילת הכפתור השני לזה של הלחיצה. אם הוא תואם – ממשיך ל-afterCheatsX2. אם לא – עובר ל-badCheatsMPos.
afterCheatsX2	משווה את פרמטר ה-x של הלחיצה לסוף הכפתור השני לזה של הלחיצה. אם הוא תואם – ממשיך ל-afterCheatsY21. אם לא – עובר ל-badCheatsMPos .
checkCheatsMouseY11	משווה את פרמטר ה-y של הלחיצה לתחילתו של הכפתור הראשון. אם הם תואמים – ממשיך ל-checkCheatsMouseY12. אם הם לא תואמים, עובר ל-badCheatsMPos.
checkCheatsMouseY12	משווה את פרמטר ה-y של הלחיצה לסופו של הכפתור הראשון. אם הם תואמים – ממשיך ל-cheatsStart. אם הם לא תואמים, עובר ל-badCheatsMPos.
checkCheatsMouseY21	משווה את פרמטר ה-y של הלחיצה לתחילתו של הכפתור השני. אם הם תואמים – ממשיך ל-checkCheatsMouseY22. אם הם לא תואמים, עובר ל-badCheatsMPos.
checkCheatsMouseY22	משווה את פרמטר ה-y של הלחיצה לסופו של הכפתור השני. אם הם תואמים – ממשיך ל-cheatsCharacter. אם הם לא תואמים, עובר ל-badCheatsMPos.
cheatsStart	מסמן כי כפתור שינוי השלב ההתחלתי נלחץ וחוזר לקריאה המקורי. (ret)
cheatsCharacter	מסמן כי כפתור שינוי "דמות" השחקן נלחץ וחוזר לקריאה המקורי. (ret)
badCheatsMPos	מאפס את הסמנים checkCheatStart, checkCheatCharacter המסמנים האם הלחיצה היתה על "כפתור".

פרוצדורות אלו אחראיות על ניהול קלט העכבר במסכים start, cheats. כל לחיצה מבוססת על אותו רעיון בסיסי:



פרוצדורות עזר לשינוי מיקום השחקן

checkPos	משווה את הערך שבמיקום ה-cursor לרווח. אם שווה חוזר לפעולה שקראה לו. אם לא שווה, עובר לפעולה badPos.
badPos	משווה את הערך שבמיקום ה-cursor ל-'*'. אם הוא שווה, עובר לפרוצדורה resetMazePos. אחרי כן משווה את הערך שבמיקום ה-cursor ל-'«' (ALT-173). אם הוא שווה, עובר לפרוצדורה newLevel. אחר כך משווה את הערך שבמיקום ה-cursor ל-'Φ' (ALT-232). אם הוא שווה, עובר לפרוצדורה win. לאחר מכן (אם הערך שבמיקום ה-cursor לא שווה לאף אחד מהסימנים האלה, כלומר קיר) מחזיר את ה-cursor למיקום ע"פ המשתנים x, y ומדפיס את השחקן. אחרי כן עובר ל-posChange.
resetCursor	מדפיס ' ' (רווח) במיקום ה-cursor.

פרוצדורות אלו נקראות מהפרוצדורות שמזיזות את השחקן (posChangeS וכו'). הן משמשות כעזר ובסופו של דבר חוזרות לפעולה המקורית שקראה להם.

פרוצדורות לניהול המסך וה-video mode

clearScreen	מגדיר את ה-video mode מחדש (מאפס את המסך) ואז חוזר לקריאה האחרונה אליו (ret).
win	קורא ל-clearScreen, מדפיס את מסך הנצחון ואז עובר ל-exitGame.
exitGame	מקבל קלט מהעכבר, וכאשר נלחץ המקש השמאלי עובר לפרוצדורה exit.
error	קורא ל-clearScreen, מדפיס את מסך ה-error ואז עובר לפרוצדורה waitForClick.
waitForClick	מקבל קלט מהעכבר, וכאשר נלחץ המקש השמאלי עובר לפרוצדורה restart.
restart	מאפס את המשתנים הסופרים את השלבים וחוזר ל-beginning.

פרוצדורות אלו משמשות לאיפוס המסך ולהדפסת מסכים שונים.

פרוצדורות לניהול שלבים

firstLevel	מאפס את המסך ואז מדפיס את השלב הראשון (ע"פ mazesIndex) ואז חוזר לפרוצדורה שממנה הוא נקרא (ret).
nextLevel	מאפס את המסך ומעלה את mazeCount ב-1 ואז קורא ל-whichMaze. כאשר whichMaze מסיים וחוזר הפרוצדורה מדפיסה את השלב הבא (ע"פ mazesIndex).
newLevel	קורא ל-nextLevel, ל-newMazePos, ול-output ואז עובר ל-posChange.
newMazePos	מאפס את x, y, xCheck, yCheck ואת מיקום ה-cursor ל-(x=0; y=1) ואז חוזר לפרוצדורה שממנו הוא נקרא (ret).
resetMazePos	קורא ל-newMazePos ול-ouput ואז עובר ל-posChange.

פרוצדורות לחישוב השלב הבא

whichMaze	משווה את הערך הנמצא במערך mazes במיקום ע"פ הערך mazesIndex ל-'$'. אם הוא שווה, עובר ל-endofMaze. אם לא שווה, עובר ל-nextChar.
endOfMaze	מעלה את ערך המשתנה $Count ב-1 ואז משווה את הערך של mazeCount לערך של $Count. אם שווה, עובר לפרוצדורה fin. אם לא שווה, עובר לפרוצדורה nextChar.
nextChar	מעלה את ערך המשתנה mazesIndex ואז עובר ל-whichMaze.
fin	מעלה את ערך המשתנה mazesIndex וחוזר לקריאה האחרונה (ret).

פעולות אלו הן בעצם לולאה (מתחיל ב-whichMaze, עובר ל-endOfMaze/nextChar לפי תנאי ומסתיים ב-fin).

העקרון עליו מתבססת הלולאה מתחלק ל-2:

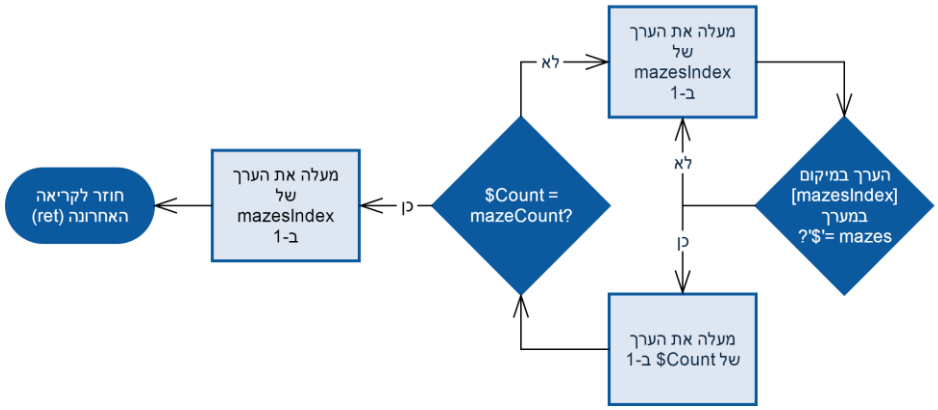
- הדרך שבה מאוחסנת מחרוזת בזכרון – מערך שבו כל אות מאוחסנת כערך ASCII בתא נפרד.
- הפסיקה 'AH 9, INT 21h' עוברת על המחרוזת – המערך ומדפיסה כל תא עד שהתא מכיל את הסימן '$'.

הלולאה עוברת על כל תא במערך mazes (מתחילה במיקום ע"פ mazesIndex ומעלה את ערכו ב-1 לאחר בדיקת התא) עד שהיא פוגשת בתא המכיל '$', כלומר סוף של שלב, ואז מעלה את מספר השלבים שכבר נספרו במערך, $Count, ומשווה את זה למספר השלבים שהשחקן עבר, mazeCount, (או מספר השלבים שהשחקן רוצה לדלג עליהם במקרה שהשחקן בחר לשנות את שלב ההתחלה במסך ה-cheats).

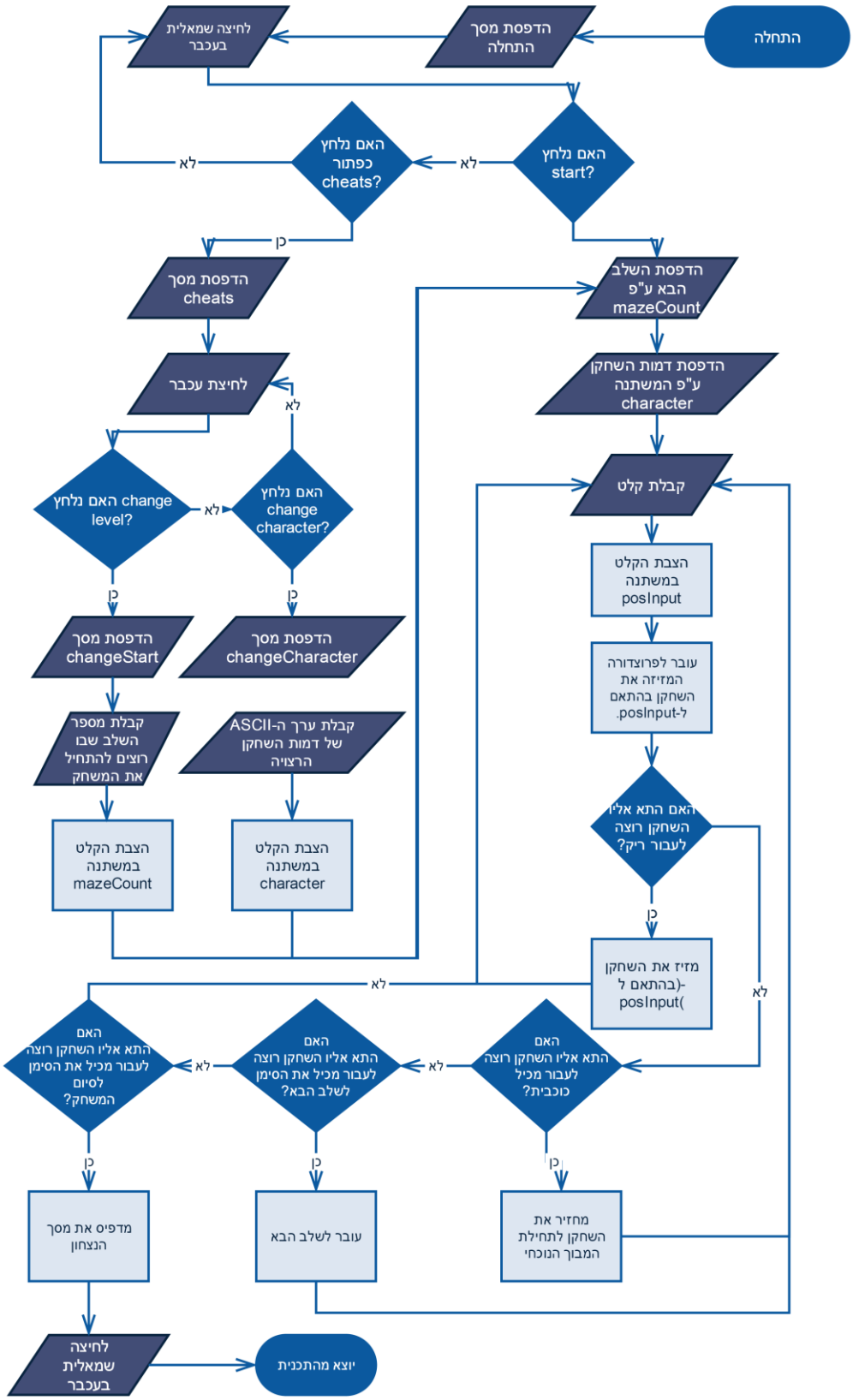
אם מספר השלבים שנספרו שווה למספר השלבים שצריך לספור (mazeCount) הלולאה מעלה את הערך של mazesIndex וחוזרת לפעולה שממנה נקראה (ret).

אם מספר השלבים שנספרו לא שווה למספר השלבים שצריך לספור (mazeCount) הלולאה מעלה את הערך של mazesIndex וחוזרת לספירה (חזרה ל-whichMazes).

אפשר לתאר זאת בתרשים זרימה:



תרשים זרימה כללי



סיכום

בתחילת השנה, אם הייתי נדרש לבנות משחק כזה לא הייתי יודע מאיפה להתחיל, שלא לדבר על שימוש בעכבר, הזזה של דמויות או אפילו מסך פתיחה!

אך לאחר הרבה למידה עצמית, קריאה באינטרנט וניסוי וטעייה הצלחתי לבנות משחק שאפשר להתגאות בו.

משחק זה מדגים בצורה יסודית את אחד מהעקרונות של השפה Assembly 8086 ושל עוד דברים רבים בתכנות המודרני, והוא הצורה שבה מאוחסנות מחרוזות בזכרון המעבד. עקרון זה מובע באמצעות הלולאה המתוארת בעמוד 8.

ביבליוגרפיה

- http://www.reddit.com/r/learnprogramming
- http://stackoverflow.com/
- http://en.wikipedia.org/wiki/Text_mode#PC_common_text_modes

קוד התכנית

```
data segment

    mazes       db "----------------------------------------------|", 0Ah, 0Dh
                db "   |*              |*        |      |      |      |", 0Ah, 0Dh
                db "|  |------------  |---  |  |  |  |  |  |  |  |", 0Ah, 0Dh
                db "|  |*      *|            |  | *|      |      | *|", 0Ah, 0Dh
                db "|  |--|  ---|  |  |-----|  |------  |-----|  |", 0Ah, 0Dh
                db "|     |        |  |     |  |        |        |", 0Ah, 0Dh
                db "|---* |  |-----|  |  |  |  |  ------|  ------|", 0Ah, 0Dh
                db "|        |        |  |* |  |        |*    |  |", 0Ah, 0Dh
                db "|  ------|  ------|  |--| *|  |---  |--- *|  |", 0Ah, 0Dh
                db "|       *|       *|           |*    |        »", 0Ah, 0Dh
                db "|----------------------------------------------|$",

                db "-----------------------------------------------", 0Ah, 0Dh
                db "          |*                |*    | *|        |", 0Ah, 0Dh
                db "|  ----------------|  |-----|  |--|  |  |  |* |", 0Ah, 0Dh
                db "|                    |        |     |* |  |  |", 0Ah, 0Dh
                db "|--| *----------* |  | *-------  |--|  |  | *|", 0Ah, 0Dh
                db "|* |     |*       |  |              |  *  |  |", 0Ah, 0Dh
                db "|  |---  |  |---  |  |  |--------|  |  |--|* |", 0Ah, 0Dh
                db "|       *|  |    *|     |        |    *|* |  |", 0Ah, 0Dh
                db "|  ---|  |--|  |--------|  | *-------  |  | *|", 0Ah, 0Dh
                db "|    *|        |*          |            |     »", 0Ah, 0Dh
                db "----------------------------------------------|$",

                db "----------------------------------------------|", 0Ah, 0Dh
                db "   |*       |        | *|        |*          |", 0Ah, 0Dh
                db "|* |  |---* |  |--|  |  |  |---  |  |-----| *|", 0Ah, 0Dh
                db "|    *|*       | *|  |     |*    | *|     |  |", 0Ah, 0Dh
                db "|-----|  |  ---|  |  |* |--|  -------* |* |* |", 0Ah, 0Dh
                db "|*      *|*    |  |     |*              |     |", 0Ah, 0Dh
                db "|--------|  |* |  |  |--------------- *| *---|", 0Ah, 0Dh
                db "|* |        |  |     |*                |    *|", 0Ah, 0Dh
                db "|  | *|  |--| *|-----|  -------------------  |", 0Ah, 0Dh
                db "|     |  |*    |*    |*         *|           |", 0Ah, 0Dh
                db "|-----|  |  -------  |  |--------|* |-----|* |", 0Ah, 0Dh
                db "|*       |  |*       | *|*          |    *|  |", 0Ah, 0Dh
                db "|  |-----|  |  |-----|  |  ---------|  |  |--|", 0Ah, 0Dh
                db "|  |*    |     |    *|* |*      *|    *|     |", 0Ah, 0Dh
                db "|------ *|-----|  |  |  |------  |* |------ *|", 0Ah, 0Dh
                db "|*               *|    *|*          |*       Φ", 0Ah, 0Dh
                db "|----------------------------------------------|$"

    victory     db "  __       __          _                    _ ", 0Ah, 0Dh
                db " \ \     / /(_) ___| |_ ___  _ __ _    _| |", 0Ah, 0Dh
                db "  \ \   / / | |/ __| __/ _ \| '__| | | | |", 0Ah, 0Dh
                db "   \ \/ /  | | (__| || (_) | |  | |_| |_|", 0Ah, 0Dh
                db "    \__/   |_|\___|\__\___/|_|   \__, (_)", 0Ah, 0Dh
                db "                                  __/ /   ", 0Ah, 0Dh
                db "                                 |___/    ", 0Ah, 0Dh
                db "                                          ", 0Ah, 0Dh
                db "         left click to exit game         $"
```

```
titleScreen db "  __   __                          ", 0Ah, 0Dh
            db " |  \/  |                         ", 0Ah, 0Dh
            db " | .  . | __ _ _______  ___       ", 0Ah, 0Dh
            db " | |\/| |/ _` |_  / _ \/ __|      ", 0Ah, 0Dh
            db " | |  | | (_| |/ /  __/\__ \      ", 0Ah, 0Dh
            db " |_|  |_|\__,_/___\___||___/      ", 0Ah, 0Dh
            db "                                  ", 0Ah, 0Dh
            db "        _______________           ", 0Ah, 0Dh
            db "        |             |           ", 0Ah, 0Dh
            db "        |     play    |           ", 0Ah, 0Dh
            db "        |     game    |           ", 0Ah, 0Dh
            db "        |_____________|           ", 0Ah, 0Dh
            db "                                  ", 0Ah, 0Dh
            db "        >press for cheats       $"

cheatScreen db "  _____  _                  _     ", 0Ah, 0Dh
            db "/  __ \| |                | |    ", 0Ah, 0Dh
            db "| /  \/| |__   ___  __ _| |_ ___ ", 0Ah, 0Dh
            db "||     | '_ \ / _ \/ _` | __/ __|", 0Ah, 0Dh
            db "| \__/\| | | |  __/ (_| | |_\__ \", 0Ah, 0Dh
            db " \____/|_| |_|\___|\__,_|\__|___/", 0Ah, 0Dh
            db "                                  ", 0Ah, 0Dh
            db "            change level          ", 0Ah, 0Dh
            db "                                  ", 0Ah, 0Dh
            db "          change character        ", 0Ah, 0Dh
            db "                                 $"

changeStartScreen     db "enter starting level: $"

changeCharacterScreen db "enter character: ", 0Ah, 0Dh
                      db "                 ", 0Ah, 0Dh
                      db "choose from: (1)", 0Ah, 0Dh
                      db "              (2)", 0Ah, 0Dh
                      db "              (3)", 0Ah, 0Dh
                      db "              (4)", 0Ah, 0Dh
                      db "              (5)", 0Ah, 0Dh
                      db "              (6)$"

errorScreen db "                    __---__                       ", 0Ah, 0Dh
            db "                  _-       \--_________           ", 0Ah, 0Dh
            db "             __--( /     \ )XXXXXXXXXXX\         ", 0Ah, 0Dh
            db "          .-XXX(   O   O  )XXXXXXXXXXXXXXX\      ", 0Ah, 0Dh
            db "         /XXX(       U     )        XXXXXXX\     ", 0Ah, 0Dh
            db "       /XXXXX(              )--_  XXXXXXXXXXX\   ", 0Ah, 0Dh
            db "      /XXXXX/ (      O     )   XXXXXX   \XXXXX\ ", 0Ah, 0Dh
            db "      XXXXX/   /            XXXXXX   \__ \XXXXX ", 0Ah, 0Dh
            db "      XXXXXX__/          XXXXXX         \__---->", 0Ah, 0Dh
            db "____  XXX__/          XXXXXX      \__          / ", 0Ah, 0Dh
            db "\-  --__/   ___/\  XXXXXX            /  ___--/X ", 0Ah, 0Dh
            db " \-\     ___/    XXXXXX              '--- XXXXXX ", 0Ah, 0Dh
            db "    \-\/XXX\ XXXXXX                      /XXXXX ", 0Ah, 0Dh
            db "      \XXXXXXXXX   \                    /XXXXX/ ", 0Ah, 0Dh
            db "       \XXXXXX      >                  _/XXXXX/  ", 0Ah, 0Dh
            db "         \XXXXX--__/              __-- XXXX/    ", 0Ah, 0Dh
            db "          \XXXXXXXX---------------  XXXXXX/     ", 0Ah, 0Dh
            db "             \XXXXXXXXXXXXXXXXXXXXXXXXX/        ", 0Ah, 0Dh
            db "               \XXXXXXXXXXXXXXXXXX/              ", 0Ah, 0Dh
```

```
                db "                 _____                              ", 0Ah, 0Dh
                db "               |  ___|                             ", 0Ah, 0Dh
                db "               | |__   _ __ _ __ ___   _ __         ", 0Ah, 0Dh
                db "               |  __|| '__| '__/  _ \| '__|        ", 0Ah, 0Dh
                db "               | |__ | |   | | | (_) | |           ", 0Ah, 0Dh
                db "               |____||_|   |_|  \___/|_|           ", 0Ah, 0Dh
                db "                 You pressed a bad key!            ", 0Ah, 0Dh
                db "                   click to restart              $"

    x                   db 0h
    y                   db 0h
    xCheck              db 0h
    yCheck              db 0h
    posInfo             db 0h
    posInput            db 0h

    mouseX              dw 0h
    MouseY              dw 0h
    mouseClick          dw 0h
    startMPos           db 0h

    checkCheatStart     dw 0h
    checkCheatCharacter dw 0h

    mazesIndex          dw 0h
    mazeCount           dw 0h
    $Count              dw 0h

    character           db 01h

ends

stack segment

    dw   128  dup(0)

ends

code segment

    start:

    mov  ax, data
    mov  ds, ax
    mov  es, ax

    beginning:

    call clearScreen

    lea  dx, titlescreen
    mov  ah, 9
    int  21h

    startMouse:

    mov  ax, 3h
```

```
int  33h
mov  mouseClick, bx
mov  mouseX, cx
mov  mouseY, dx

call checkStartMClick

cmp  startMPos, 1h
je   goodStartMPos
jne  badStartMPos

cheats:

call clearScreen

lea  dx, cheatScreen
mov  ah, 9h
int  21h

cheatsMouse:

mov  ax, 3h
int  33h
mov  mouseClick, bx
mov  mouseX, cx
mov  mouseY, dx

call checkCheatsMClick

cmp  checkCheatStart, 1h
je   changeStart

cmp  checkCheatCharacter, 1h
je   changeCharacter

changeStart:

call clearScreen

lea  dx, changeStartScreen
mov  ah, 9h
int  21h

mov  dl, 16h
mov  dh, 0h
mov  bh, 0h
mov  ah, 2h
int  10h

mov  ah, 1h
int  21h

sub  al, 32h
mov  byte ptr mazeCount, al
call newLevel
```

```
changeCharacter:

call clearScreen

lea  dx, changeCharacterScreen
mov  ah, 9h
int  21h

mov  dl, 11h
mov  dh, 0h
mov  bh, 0h
mov  ah, 2h
int  10h

mov  ah, 1h
int  21h

sub  al, 30h
mov  character, al
jmp  goodStartMPos

goodStartMPos:

call firstLevel
call newMazePos
call output

posChange:

call invisInput
cmp  posInput, '1'
je   posChangeSW
cmp  posInput, '2'
je   posChangeS
cmp  posInput, '3'
je   posChangeSE
cmp  posInput, '4'
je   posChangeW
cmp  posInput, '6'
je   posChangeE
cmp  posInput, '7'
je   posChangeNW
cmp  posInput, '8'
je   posChangeN
cmp  posInput, '9'
je   posChangeNE
cmp  posInput, '5'
je   error

exit:

call clearScreen

mov  ax, 4c00h
int  21h
```

```
invisInput:  mov  ah, 7h
             int  21h
             mov  posInput, al
             ret

output:      mov  dl, character
             mov  ah, 2h
             int  21h
             ret

checkStartMClick:

             cmp  mouseClick, 1h
             je   checkMouseX
             jne  startMouse

checkMouseX: cmp  mouseX, 36h
             jae  afterStartX
             jb   startMouse

afterStartX: cmp  mouseX, 0B8h
             jbe  goodStartXPos
             ja   checkCheatX

goodStartXPos:

             mov  startMPos, 1h
             jmp  checkStartMouseY

checkStartMouseY:

             cmp  mouseY, 36h
             jae  afterStartY
             jb   startMouse

afterStartY: cmp  mouseY, 58h
             jbe  goodStartYPos
             ja   checkCheatY1

goodStartYPos:
             mov  startMPos, 1h
             ret

checkCheatX: cmp  mouseX, 0C0h
             jbe  cheats
             ja   badStartMPos

checkCheatY1:cmp  mouseY, 60h
             jae  checkCheatY2
             jb   badStartMPos

checkCheatY2:cmp  mouseY, 68h
             jbe  cheats
             ja   badStartMPos
```

```
badStartMPos:mov  startMPos, 0h
             jmp  startMouse

checkCheatsMClick:

             cmp  mouseClick, 1h
             je   checkCheatsMouseX1
             jne  cheatsMouse

checkCheatsMouseX1:

             cmp  mouseX, 60h
             jae  afterCheatsX1
             jb   checkCheatsMouseX2

afterCheatsX1:

             cmp  mouseX, 0B8h
             jbe  checkCheatsMouseY11
             ja   checkCheatsMouseX2

checkCheatsMouseX2:

             cmp  mouseX, 50h
             jae  afterCheatsX2
             jb   badCheatsMPos

afterCheatsX2:

             cmp  mouseX, 0D0h
             jbe  checkCheatsMouseY21
             ja   badCheatsMPos

checkCheatsMouseY11:

             cmp  mouseY, 03Ah
             jae  checkCheatsMouseY12
             jb   badCheatsMPos

checkCheatsMouseY12:

             cmp  mouseY, 40h
             jbe  cheatsStart
             ja   checkCheatsMouseY21

checkCheatsMouseY21:

             cmp  mouseY, 48h
             jae  cheatsCharacter
             jb   badCheatsMPos

checkCheatsMouseY22:

             cmp  mouseY, 50h
             jbe  cheatsCharacter
             ja   badCheatsMPos
```

```
cheatsStart: mov  checkCheatStart, 1h
             ret

cheatsCharacter:

             mov  checkCheatCharacter, 1h
             ret

badCheatsMPos:
             mov  checkCheatStart, 0h
             mov  checkCheatCharacter, 0h
             jmp  cheatsMouse

checkPos:    mov  dl, xCheck
             mov  dh, yCheck
             mov  ah, 2h
             int  10h
             mov  bh, 0h
             mov  al, 0h
             mov  ah, 08h
             int  10h
             mov  posInfo, al
             cmp  posInfo, ' '
             jne  badPos
             ret

badPos:      cmp  posInfo, '*'
             je   resetMazePos

             cmp  posInfo, ''
             je   newLevel

             cmp  posInfo, 'Φ'
             je   win

             mov  dl, x
             mov  dh, y
             mov  ah, 2h
             int  10h
             mov  xCheck, dl
             mov  yCheck, dh
             call output
             jmp  posChange

resetCursor: mov  dl, x
             mov  dh, y
             mov  ah, 2h
             int  10h
             mov  dl, ' '
             mov  ah, 2h
             int  21h
             ret
```

```
clearScreen: mov  al, 3h
             mov  ah, 0h
             int  10h
             ret

win:         call clearScreen

             lea  dx, victory
             mov  ah, 9h
             int  21h
             jmp  exitGame

exitGame:    mov  ax, 3h
             int  33h
             mov  mouseClick, bx

             cmp  mouseClick, 01h
             je   exit
             jne  exitGame

error:       call clearScreen

             lea  dx, errorScreen
             mov  ah, 9h
             int  21h

             jmp  waitForClick

waitForClick:mov  ax, 3h
             int  33h
             mov  mouseClick, bx

             cmp  mouseClick, 01h
             je   restart
             jne  waitForClick

restart:     mov  mazesIndex, 0h
             mov  mazeCount, 0h
             mov  $Count, 0h

             jmp  beginning

firstLevel:  call clearScreen

             mov  si, mazesIndex
             lea  dx, mazes[si]
             mov  ah, 9h
             int  21h
             mov  si, 0h

             ret

nextLevel:   call clearScreen

             inc  mazeCount
             call whichMaze
```

```
            mov  si, mazesIndex
            lea  dx, mazes[si]
            mov  ah, 9h
            int  21h
            mov  si, 0h
            ret

newLevel:   call nextLevel
            call newMazePos
            call output
            jmp posChange

newMazePos: mov  xCheck, 0h
            mov  yCheck, 1h
            mov  x, 0h
            mov  y, 1h
            mov  dl, x
            mov  dh, y
            mov  bh, 0h
            mov  ah, 2h
            int  10h
            ret

resetMazePos:call newMazePos
            call output
            jmp  posChange

whichMaze:  mov  si, mazesIndex
            mov  bl, mazes[si]
            cmp  bl, '$'
            je   endOfMaze
            jne  nextChar

endOfMaze:  inc  $Count
            mov  dx, mazeCount
            cmp  $Count, dx
            je   fin
            jne  nextChar

nextChar:   inc  mazesIndex
            jmp  whichMaze

fin:        inc  mazesIndex
            ret

posChangeSW: call resetCursor

            dec  xCheck
            inc  yCheck
            call checkPos

            dec  x
            inc  y
            mov  dl, x
            mov  dh, y
            mov  ah, 2h
            int  10h
```

```
            call output
            jmp  posChange

posChangeS: call resetCursor

            inc  yCheck
            call checkPos

            inc  y
            mov  dl, x
            mov  dh, y
            mov  ah, 2h
            int  10h
            call output
            jmp  posChange

posChangeSE: call resetCursor

            inc  xCheck
            inc  yCheck
            call checkPos

            inc  x
            inc  y
            mov  dl, x
            mov  dh, y
            mov  ah, 2h
            int  10h
            call output
            jmp  posChange

posChangeW: call resetCursor

            dec  xCheck
            call checkPos

            dec  x
            mov  dl, x
            mov  dh, y
            mov  ah, 2h
            int  10h
            call output
            jmp  posChange

posChangeE: call resetCursor

            inc  xCheck
            call checkPos

            inc  x
            mov  dl, x
            mov  dh, y
            mov  ah, 2h
            int  10h
            call output
            jmp  posChange
```

```
posChangeNW: call resetCursor

             dec  xCheck
             dec  yCheck
             call checkPos

             dec  x
             dec  y
             mov  dl, x
             mov  dh, y
             mov  ah, 2h
             int  10h
             call output
             jmp  posChange

posChangeN:  call resetCursor

             dec  yCheck
             call checkPos

             dec  y
             mov  dl, x
             mov  dh, y
             mov  ah, 2h
             int  10h
             call output
             jmp  posChange

posChangeNE: call resetCursor

             inc  xCheck
             dec  yCheck
             call checkPos

             inc  x
             dec  y
             mov  dl, x
             mov  dh, y
             mov  ah, 2h
             int  10h
             call output
             jmp  posChange


ends

end start
```