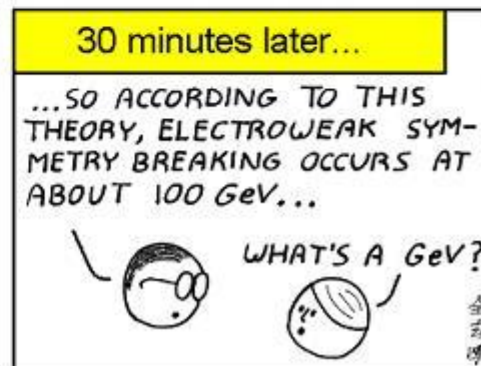
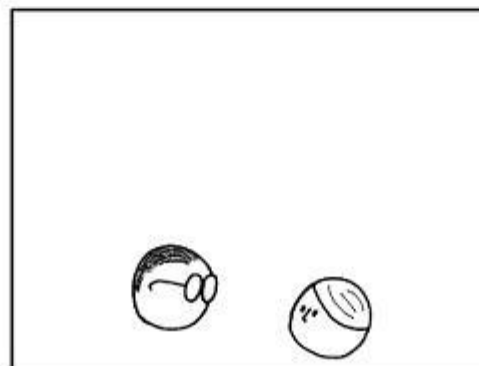
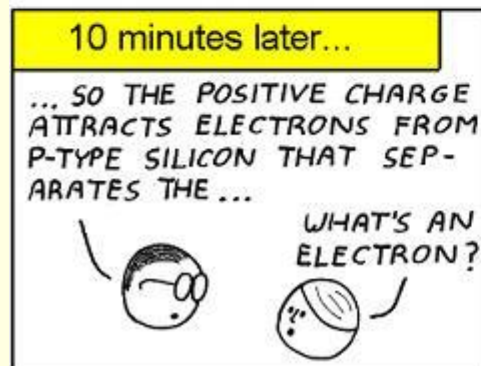
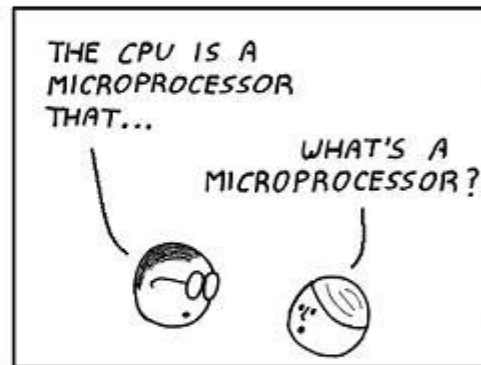
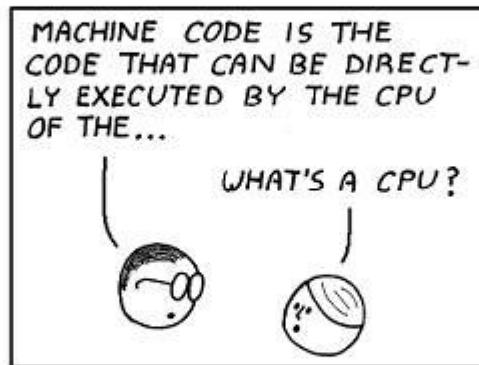
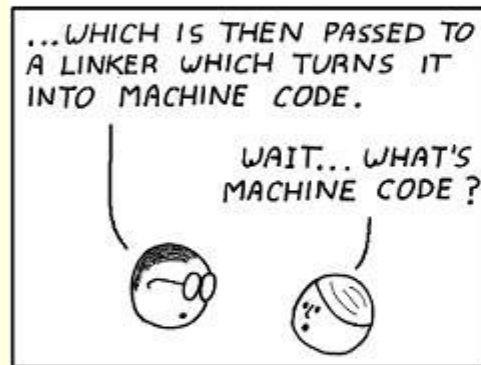
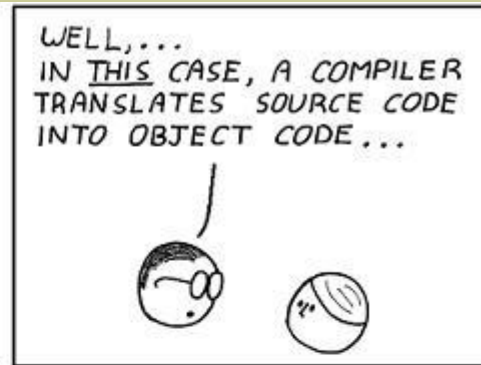
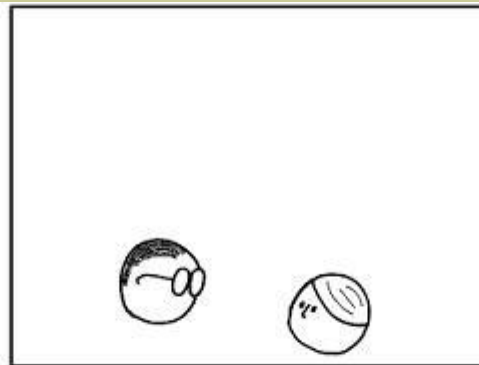
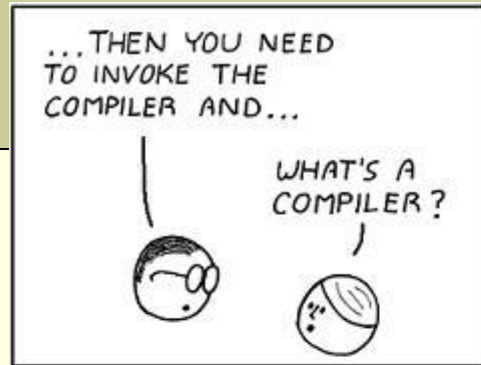


קורס תכנות

שיעור שמיני: סיכום ביניים

- מתחת למכסה המנוע
 - ניסיון להבין מעט טוב יותר את סביבת הפיתוח
- בעיית צעדי הפרש
 - רקורסיה ומערכים דו-ממדיים

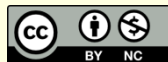


HOW DOES COMPUTER PROGRAMMING WORK?

MAGIC.



全
方
漫



Abstruse Goose, [Under The Hood](#)

אז מה היה לנו

- סביבת העבודה (חומרה, קוד, IDE)
- משתנים
- בקרת זרימה (if-else, לולאות, ...)
- פונקציות
- מערכים ומחרוזות

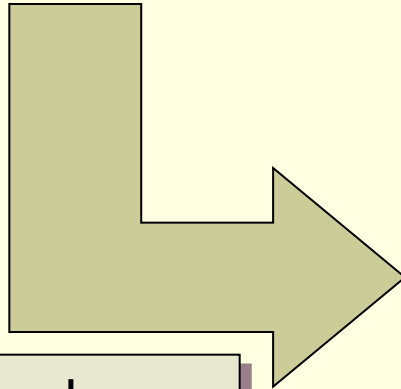
ומה עוד לפנינו?

- מצביעים וכתובות
- חיפוש מיון וסיבוכיות חישוב
- מבנים
- הקצעת זכרון דינמית
- רשימות מקושרות
- בונוס

תכנית מחשב

```
#include <stdio.h>
/* The simplest C Program */
int main()
{
    printf("Hello world\n");
    return 0;
}
```

שפה עילית (C, Java, Python, ...)



קומפיילר -
תרגום משפה עילית
לשפת מכונה

```
; Hello world for Intel Assembler
mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello

int 21h
xor ax,ax
int 21h

Hello:
    db "Hello world!",13,10,"$"
```

שפת מכונה (כאן אסמבלי)

מכתיבת הקוד לפלט

- קומפילציה (סטטי)

1. preprocessor – שינויים בטקסט של התכנית
2. compiler – תרגום משפה עילית לשפת מכונה
3. linker – חיבור של כל הרכיבים לאפליקציה אחת

- הרצה (דינאמי)

- סדרתי - פקודה אחרי פקודה


```
int main()
{
    const char rhyme[] = "Humpty Dumpty sat on a wall,\n"
                        "Humpty Dumpty had a great fall.\n"
                        "All the king's horses,\n"
                        "And all the king's men,\n"
                        "Couldn't put Humpty together again.\n";
    const char humpty[] = "Humpty";
    int index = 0;
    int count = 0;

    /* count occurrences of humpty in rhyme */
    for (index = my_strstr(rhyme, humpty, index);
         index != ERROR;
         index = my_strstr(rhyme, humpty, index + 1))
    {
        count++;
    }

    printf("The string %s appears %d times\n", humpty, count);

    return 0;
}
```

דוגמא (המשך)

```
int my_strstr(const char haystack[], const char needle[], int offset)
{
    int needlelen;

    if (needle[0] == '\0')
        return 0;

    needlelen = my_strlen(needle);
    offset = my_strchr(haystack, offset, needle[0]);
    while (offset != -1) {
        if (my_strncmp(haystack, offset, needle) == 0)
            return offset;
        offset++;
        offset = my_strchr(haystack, offset, needle[0]);
    }

    return ERROR;
}
```

ה Preprocessor

- מבצע שינויים בטקסט של התכנית
 - מתעלם מהתחביר של C
- פקודות ל preprocessor
 - `#include` – הכלה של קובץ
 - `#define` - הגדרה של מקרו (שם סימבולי)

הקוד לאחר ה Preprocessor

example.i •

```
int main()
{
    const char rhyme[] = "Humpty Dumpty sat on a wall,\n"
        ...
        "Couldn't put Humpty together again.\n";
    const char humpty[] = "Humpty";
    int index = 0;
    int count = 0;

    for (index = my_strstr(rhyme, humpty, index);
        index != -1;
        index = my_strstr(rhyme, humpty, index + 1))
    {
        count++;
    }

    printf("The string %s appears %d times\n", humpty, count);

    return 0;
}
```

ההערה נמחקה

ERROR הוחלף ב -1

הקומפיילר

- מתרגם קובץ קוד יחיד מ C לשפת מכונה
- example.cod

- גוף הלולאה – הפקודה ++count

<pre>. 73 . 74 שפת מכונה</pre>		טען את הערך של המשתנה count לרגיסטר eax
<pre>000c88b 85 3c ff ff ff 000ce83 c0 01 000d189 85 3c ff ff ff</pre>	<pre>mov eax, DWORD PTR [PT add eax, 1 mov DWORD PTR [PT</pre>	הוסף את הקבוע 1 לערך ברגיסטר eax
<pre>; 75 : } }</pre>		כתוב את הערך השמור ברגיסטר eax למקום השמור למשתנה count

ה Linker

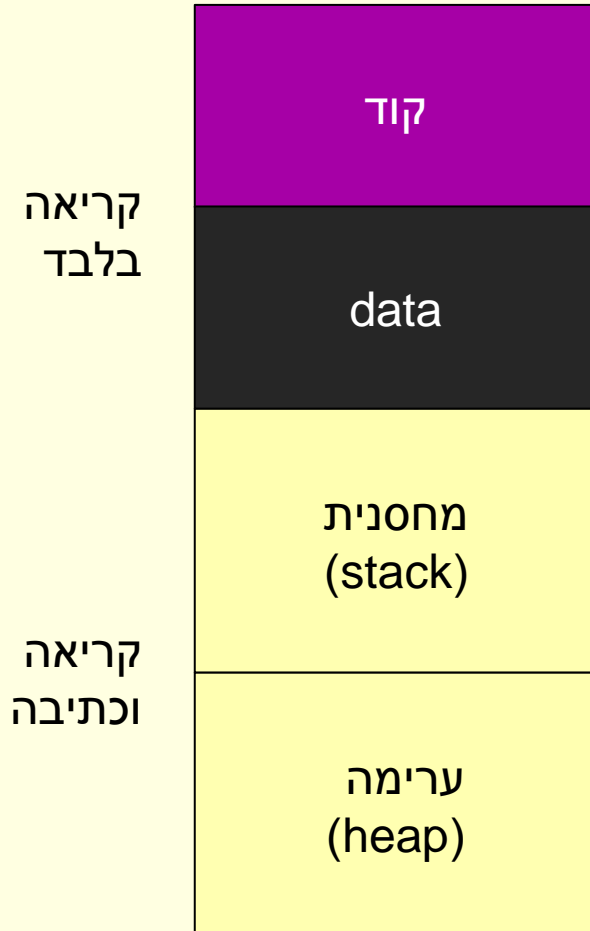
- יוצר קובץ הניתן להרצה (executable) מהפלט של הקומפיילר ומספריות Buildlog.htm •

```
1>Linking...
1>Starting pass 1
1>Searching libraries
1>    Searching C:\Program Files\Microsoft SDKs\windows\v6.0A\lib\kernel32.lib:
...
1>    Searching C:\Program Files\Microsoft Visual Studio 9.0\VC\lib\MSVCRTD.lib:
1>        Found __imp__strncmp
1>        Referenced in example.obj
1>        Loaded MSVCRTD.lib(MSVCR90D.dll)
1>        Found __imp__strchr
1>        Referenced in example.obj
1>        Loaded MSVCRTD.lib(MSVCR90D.dll)
1>        Found _strlen
1>        Referenced in example.obj
1>        Loaded MSVCRTD.lib(MSVCR90D.dll)
1>        Found __imp__printf
1>        Referenced in example.obj
```

הרצת התכנית - שלבים

1. טעינת התכנית מהדיסק לזיכרון
 - הקוד לחלק הקוד, המחרוזות הקבועות לחלק הקבוע
2. הקצאת Stack Frame עבור הפונקציה main
 - הקצאת מקום במחסנית עבור המשתנים המקומיים של main
3. קריאה לפונקציה main
4. ביצוע סדרתי של פקודות התכנית
 - קריאה לפונקציה – הקצאה של Stack Frame עבור הפונקציה וביצוע הקוד שלה
 - חזרה מקריאה – ביטול ה Stack Frame וחזרה להמשך ביצוע הפקודות ממקום הקריאה

מודל הזיכרון של התכנית



• הקוד

- פקודות התכנית

• Data

- מחרוזות שהוגדרו ע"י המתכנת

• מחסנית (Stack)

- משמש לאיחסון משתנים לוקאליים

- Last In First Out (LIFO)

- בשליטת התכנית

• ערימה (Heap)

- בשליטת המתכנת (בהמשך הקורס)

לפני הקריאה ל my_strstr

The screenshot shows the Microsoft Visual C++ 2010 Express IDE in debug mode. The main window displays the source code for a program named 'Humpty'. The code defines a rhyme and a string 'Humpty', then uses a loop with `my_strstr` to count occurrences of 'Humpty' in the rhyme. The `printf` statement outputs the result.

```
int main()
{
    const char rhyme[] = "Humpty Dumpty sat on a wall,\n"
                        "Humpty Dumpty had a great fall.\n"
                        "All the king's horses,\n"
                        "And all the king's men,\n"
                        "Couldn't put Humpty together again.\n";

    const char humpty[] = "Humpty";
    int index = 0;
    int count = 0;

    /* count occurrences of humpty in rhyme */
    for (index = my_strstr(rhyme, humpty, index);
         index != ERROR;
         index = my_strstr(rhyme, humpty, index + 1))
    {
        count++;
    }

    printf("The string %s appears %d times\n", humpty, count);

    return 0;
}
```

The **Locals** window at the bottom shows the following variables:

Name	Value	Type
humpty	0x0012feb3 "Humpty"	const ch
rhyme	0x0012fecc "Humpty Dumpty sat on a wall,Humpty I	const ch
index	0	int
count	0	int

The **Call Stack** window shows the current call stack:

Name	Lang
Humpty.exe!main() Line 73	C
Humpty.exe!_tmainCRTStartup() Line 555 + 0x19 bytes	C
Humpty.exe!mainCRTStartup() Line 371	C
kernel32.dll!7c917077	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	

מצב התכנית

מחסנית הקריאות:
אנחנו ב main

The screenshot displays the Visual Studio IDE with two windows open: 'Locals' and 'Call Stack'. The 'Locals' window shows the following variables:

Name	Value	Type
humpty	0x0012feb0 "Humpty"	const ch
rhyme	0x0012fec0 "Humpty Dumpty sat on a wall, Humpty []"	const ch
index	0	int
count	0	int

The 'Call Stack' window shows the following frames:

Name	Language
Humpty.exe!main() Line 73	C
Humpty.exe!_tmainCRTStartup() Line 555 + 0x19 bytes	C
Humpty.exe!mainCRTStartup() Line 371	C
kernel32.dll!7c817077()	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	

The bottom toolbar shows tabs for Autos, Locals, Threads, Modules, and Watch 1. A green arrow points to the 'Locals' tab.

משתנים מקומיים:
המשתנים שהוגדרו ב- main
שימו לב, המערך הוא בעצם הכתובת של התא הראשון במערך

ביצוע my_strstr

שימו לב למחסנית הקריאות ולערכי המשתנים

(כתובות) המערכים נשארו כשהיו
הערך של needlelen הוא "זבל"

```
int needlelen;  
  
if (needle[0] == '\0')  
    return 0;  
  
needlelen = my_strlen(needle);  
offset = my_strchr(haystack, offset, needle[0]);  
while (offset != -1) {  
    if (my_strncmp(haystack, offset, needle) == 0)  
        return offset;  
    offset++;  
    offset = my_strchr(haystack, offset, needle[0]);  
}
```

100 %

Name	Value	Type
haystack	0x0012fecc "Humpty Dumpty sat on a wall, Humpty"	const ch
needle	0x0012feb0 "Humpty"	const ch
offset	0	int
needlelen	-858993460	int

Name	Language
Humpty.exe!my_strstr(const char * haystack, const char * needle, int offset) Line 44	C
Humpty.exe!main() Line 73 + 0x1a bytes	C
Humpty.exe!_tmainCRTStartup() Line 555 + 0x19 bytes	C
Humpty.exe!mainCRTStartup() Line 371	C
kernel32.dll!7c817077()	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	

Autos Locals Threads Modules Watch 1 Call Stack Breakpoints Output

Ln 44 Col 1 Ch 1 INS

איפה אנחנו?

The screenshot shows a debugger's Call Stack window. The title bar reads "Call Stack" with standard window controls. The main area is a table with two columns: "Name" and "Language". The top row is highlighted with a yellow arrow pointing to the left, indicating the current execution point. Below it are several other frames, including calls to `my_strlen`, `my_strncmp`, `my_strstr`, `main`, and `!_tmainCRTStartup`. The bottom row shows a kernel32.dll frame with a note: "IFrames below may be incorrect and/or missing. no symbols loaded for kernel32.dll". At the bottom of the window, there are tabs for "Call Stack", "Breakpoints", and "Output". The status bar at the very bottom shows "Ln 24", "Col 1", "Ch 1", and "INS".

Name	Language
→ Humpty.exe!my_strlen(const char * str) Line 24	C
Humpty.exe!my_strncmp(const char * str1, int offset, const char * str2) Line 32 + 0x9 bytes	C
Humpty.exe!my_strstr(const char * haystack, const char * needle, int offset) Line 50 + 0x11 bytes	C
Humpty.exe!main() Line 73 + 0x1a bytes	C
Humpty.exe!_tmainCRTStartup() Line 555 + 0x19 bytes	C
Humpty.exe!mainCRTStartup() Line 371	C
kernel32.dll!7c817077()	
IFrames below may be incorrect and/or missing. no symbols loaded for kernel32.dll	

Call Stack Breakpoints Output

Ln 24 Col 1 Ch 1 INS

- **משתנים נמצאים בזיכרון בכתובת מסוימת**
 - הקומפילר יודע בדיוק מה הכתובת, אנחנו לא
- **אנחנו משתמשים בשמות של משתנים**
- **הקומפילר מתרגם את השם לכתובת**



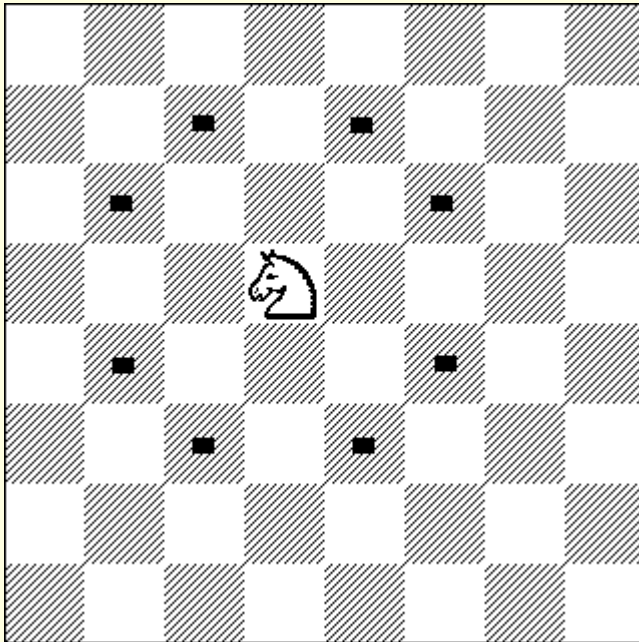
שאלות נוספות



בעיית צעדי הפרש

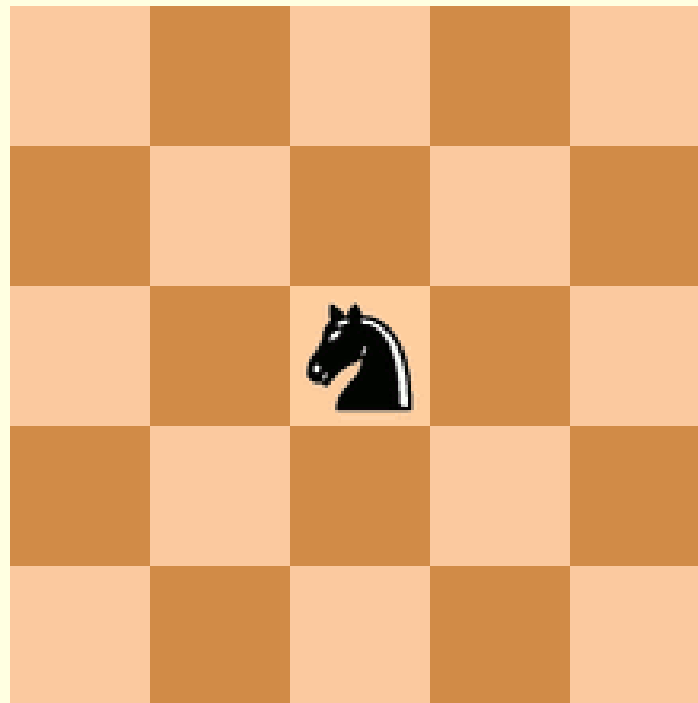
בעיית צעדי הפרש

- הקדמה למי שלא יודע שחמט:
 - הפרש הוא כלי שניתן להזיזו רק לפי הכלל הבא:
2 צעדים בכיוון מסוים, ולאחר מכן צעד בודד בכיוון מאונך לכיוון הראשוני



הבעיה

- בהינתן לוח שח ופרש, הממוקם על אחת ממשבצות הלוח, יש להגיע לכל אחת ממשבצות הלוח האחרות פעם יחידה, תוך ביצוע צעדי פרש



מאיפה מתחילים?

- מבינים את הבעיה טוב יותר
- האם אכפת לנו שמדובר בלוח שח?
 - משנה לנו מה הצבעים של המשבצות?
- מה בעצם מבקשים?
 - מסלול שמכסה את כל המשבצות ושלא חוזר על משבצת פעמיים
 - מעבר מוגבל מנקודה לנקודה

מידול עולם הבעיה

- כיצד נייצג את לוח השח?
 - מערך דו מימדי
- איך נזכור איפה ביקרנו ואיפה לא?
 - לפי הערך בתא ה- x, y
- מהם כל הצעדים האפשריים ממקום מסוים?
 - יש שמונה. לפי חוקי התנועה של הפרש
- מהו צעד חוקי?
 - כזה שנשאר בתוך הלוח ולא מבקר במקום שכבר היינו בו
- מתי נדע שמצאנו פתרון?
 - כשמספר הצעדים שביצענו מכסה את כל הלוח
- מצאנו פתרון, מהו?
 - נצטרך לחשוב על דרך להציג אותו, הדפסה למשל.

ניסיון רקורסיבי

- רקורסיה על מה?

- על המסלול

- אם אני יכול לבצע צעד אחד מה נותר לי לעשות?

- למצוא מסלול מהמקום החדש שמכסה את כל המשבצות ושלא חוזר על משבצת פעמיים ואורכו קצר

- תנאי עצירה?

- כיסינו את כל הלוח

- כדי לייעל, נזכור מה אורך המסלול עד עתה. נסיים כאשר ביצענו מספיק צעדים כדי לכסות את הלוח

אלגוריתם

```
algorithm boolean knightMove(x, y, m)
    not valid move?
        return false
    visited[x,y] ← true
    finished?
        return true
    else
        for each possible move [dx, dy]
            if knightMove(x + dx, y + dy, m + 1)
                return true
    visited[x,y] ← false
    return false
```

זה לא קוד ב-C!

אלגוריתם

```
algorithm boolean knightMove(x, y, m)
    not valid move?
        return false
    visited[x,y] ← true
    finished?
        return true
    else
        for each possible move [dx, dy]
            if knightMove(x + dx, y + dy, m + 1)
                return true
    visited[x,y] ← false
    return false
```

x, y – המשבצת על הלוח אליה
מבקשים לנוע
 m – מספר הצעדים עד כה

אלגוריתם

```
algorithm boolean knightMove(x, y, m)
    not valid move?
        return false
    visited[x,y] ← true
    finished?
        return true
    else
        for each possible move [dx, dy]
            if knightMove(x + dx, y + dy, m + 1)
                return true
    visited[x,y] ← false
    return false
```

visited – ייצוג של לוח השחמט
בכל תא נשמור האם ביקרנו
בו כבר

פתרון

מערך דו-מימדי המתאר את כל התנועות האפשריות של הפרש ביחס לנקודה הנוכחית

```
int main()
{
    int moves[][2] = { {1,2}, {2,1}, {2,-1}, {1,-2},
                       {-1,-2}, {-2,-1}, {-2,1}, {-1,2} };

    int visited[BOARD_SIZE][BOARD_SIZE];

    init_visited(visited);

    if ( !knightMove(visited, moves, 2, 2, 0) )
        printf("No solution was found!");
    printf("\n");

    return 0;
}
```


פתרון

```
int main()
{
    int moves[][2]
    int visited[BOARD_SIZE][BOARD_SIZE];

    init_visited(visited);

    if ( !knightMove(visited, moves, 2, 2, 0) )
        printf("No solution was found!");
    printf("\n");

    return 0;
}
```

מערך דו-מימדי המייצג את לוח השחמט
(TRUE/FALSE) נשמור האם בקרנו בו כבר או לא

פתרון

```
int main()
{
    int moves[][2] = { {1,2}, {2,1}, {2,-1}, {1,-2},
                      {-1,-2}, {-2,-1}, {-2,1}, {-1,2} };

    int visited[B]
    init_visited(visited);

    if ( !knightMove(visited, moves, 2, 2, 0) )
        printf("No solution was found!");
    printf("\n");

    return 0;
}
```

נשמור את הערך FALSE בכל אחד מתאי המערך (לא ביקרנו באף משבצת)

פתרון

```
int main()
{
    int moves[][2] = { {1,2}, {2,1}, {2,-1}, {1,-2},
                      {-1,-2}, {-2,-1}, {-2,1}, {-1,2} };

    int visited[BOARD_SIZE][BOARD_SIZE];

    init_visited(
        [2,2] היא לבעיה כשנקודת ההתחלה של הפרש היא 0 ומספר הצעדים שבצענו עד כה הוא 0

    if ( !knightMove(visited, moves, 2, 2, 0) )
        printf("No solution was found!");
    printf("\n");

    return 0;
}
```

פתרון

```
void init_visited(int board[][BOARD_SIZE], int size)
{
    int i, j;

    for (i = 0; i < BOARD_SIZE; i++)
        for (j = 0; j < BOARD_SIZE; j++)
            board[i][j] = FALSE;
}
```

שמור את הערך FALSE בכל אחד מתאי המערך הדו-מימדי

פתרון

```
int knightMove(int visited[][BOARD_SIZE], int moves[][2], int x, int y, int m)
{
    int i;

    if ( (x < 0) || (x >= BOARD_SIZE) || (y < 0) || (y >= BOARD_SIZE) )
        return FALSE;    // A coordinate is off the board
    if (visited[x][y])
        return FALSE;    // Can't move here; it has already been visited

    if (m == BOARD_SIZE * BOARD_SIZE - 1) { // Found solution; Start printing
        printf("A solution has been found [%d,%d] ", x , y);
        return TRUE;
    }

    visited[x][y] = TRUE;
    for (i = 0; i < 8; ++i) {
        if (knightM(visited, moves, x + moves[i][0], y + moves[i][1], m + 1)) {
            printf(" [%d,%d] ", x , y);
            return TRUE;
        }
    }
    visited[x][y] = FALSE;
    return FALSE;
}
```