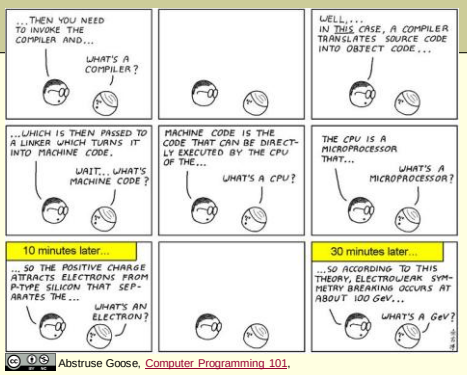
קורס תכנות

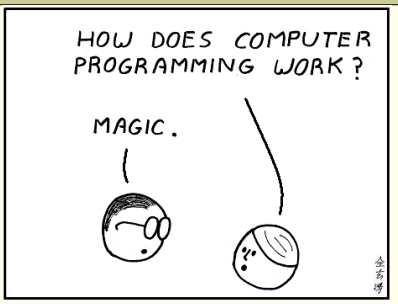
שיעור שמיני: סיכום ביניים

היום

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



Abstruse Goose, Computer Programming 101,



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

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

8

תכנית מחשב

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

שפה עילית (C, Java, Python, ...)

```
; 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,"$"
```

קומפיילר - תרגום משפה עילית לשפת מכונה

שפת מכונה (כאן אסמבלי)

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;
}
```

10

דוגמא

```
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 occurances 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;
}
```

9

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

12

ה Preprocessor

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

11

ה Linker

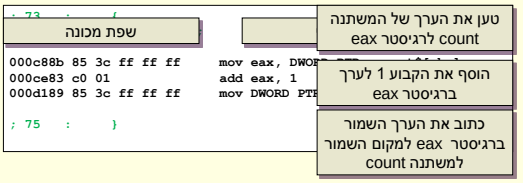
- יוצר קובץ הניתן להרצה (executable) מהפלט של הקומפיילר ומספריות
  - Buildlog.htm

```
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
```

14

הקומפיילר

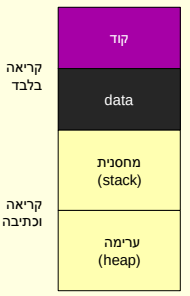
- מתרגם קובץ קוד יחיד מ C לשפת מכונה
  - example.cod
- גוף הלולאה – הפקודה ++count



13

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)



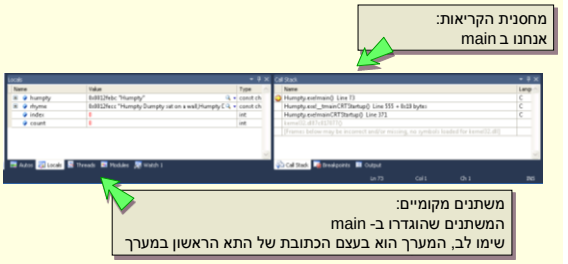
16

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

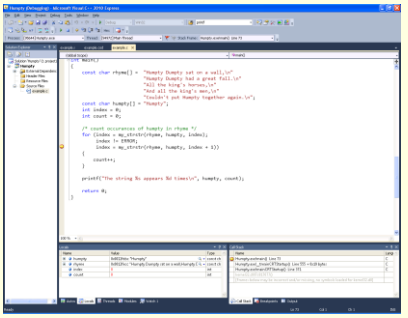
15

מצב התכנית



18

לפני הקריאה ל my_strstr

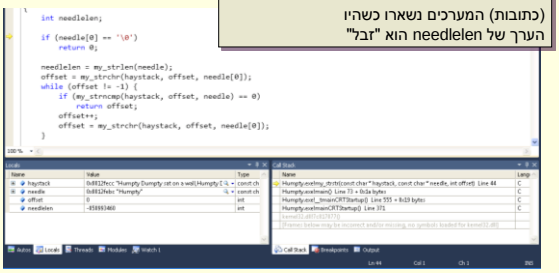


17

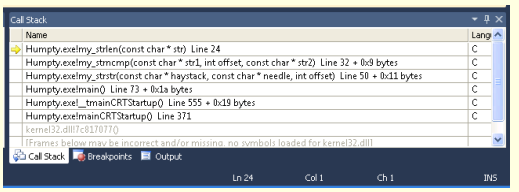
ביצוע my_strstr

שימו לב למחסנית הקריאות ולערכי המשתנים

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



איפה אנחנו?



סיכום

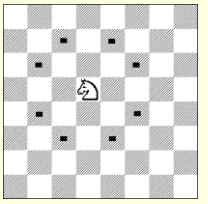
- משתנים נמצאים בזיכרון בכתובת מסוימת
  - הקומפיילר יודע בדיוק מה הכתובת, אנחנו לא
- אנחנו משתמשים בשמות של משתנים
- הקומפיילר מתרגם את השם לכתובת

שאלות נוספות

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



בעיית צעדי הפרש

הבעיה

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

25

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

26

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.

27

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

28

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!

29

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

30

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 בכל תא בלוח, נשמור האם בקרנו בו כבר או לא (TRUE/FALSE)

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

    return 0;
}
```

האם קיים פתרון לבעיה כשנקודת ההתחלה של הפרש היא [2,2] ומספר הצעדים שבצענו עד כה הוא 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;
}
```