פרק 6
לולאות

קריאה עצמית

משחק התליין (הגרסה המספרית)

- המחשב בוחר מספר אקראי **num**
- בכל סיבוב של המשחק:
  - השחקן מנחש ספרה
  - המחשב מחפש את הספרה ב-**num**
  - אם הספרה אינה קיימת, זה נספר ככישלון.
  - המחשב מדפיס את הספרות שנתגלו, והיתר מודפסות כ ' – '
- אם המספר מתגלה, או שהשחקן מגיע למספר הכישלונות המקסימלי, נגמר המשחק.

סכמה: מבנה התוכנית

```
#define TRY_NUM 5

int main()
{
  < בחר מספר אקראי num >

  revealed = 0; tries_left = TRY_NUM;
  while((tries_left > 0)  &&  (!revealed))
  {
    < קרא ספרה מהשחקן >
    < חפש את הספרה ב-num, ואם לא נמצא, tries_left-- >
    < הצג את הספרות הידועות עד כה >
    < אם כל הספרות ידועות, revealed=1 >
  }
  return 0;
}
```


בחירת המספר האקראי למשחק

```
#include <time.h>
#include <stdlib.h>

int main()
{
  unsigned int num;
  srand(time(0));
  num = rand();
}
```

הפונקציה **time()** מחזירה את הזמן הנוכחי, כמספר שלם (בד"כ מספר השניות מאז 1.1.1970 00:00).
מוגדרת בספרייה **<time.h>**.

איך נזכור את מצב המשחק?

- המחשב צריך לזכור אילו ספרות נתגלו עד כה על ידי המשתמש.
- נשתמש במשתנה נוסף שנקרא לו **known_dig** ויהיה גם כן מטיפוס **unsigned int**.
- המספר הזה יהיה מספר מיוחד, שמכיל 1 בכל מקום שבו המשתמש גילה את הספרה, ו-0 בכל מקום אחר. למשל:

המספר **num** הוא:	**27564**
הספרות שנתגלו עד כה:	**2--64**
known_dig יהיה:	**10011**

קריאת הניחוש מהשחקן

- נכתוב לולאה שקוראת מהמשתמש מספר שלם, ונמשכת עד שהמספר שנקרא הוא אכן ספרה חוקית.
- כיוון שצריך להיכנס ללולאה לפחות פעם אחת (לשם ביצוע קריאת הקלט) נשתמש בלולאת **do-while**.

```
do {
  printf("\nenter a digit (0..9): ");
  if (scanf("%d", &digit) != 1) {
     printf("error in input.. bye!\n");
     return 0;
  }
} while (digit<0 || digit>9);
```

סימון הספרות שניחש השחקן

- נסרוק את המספר הסודי ספרה-ספרה, ונשווה כל ספרה לניחוש של השחקן. אם נקבל התאמה, נעדכן את **known_dig**.
- לשם עדכון **known_dig**, נבדוק איזו ספרה יש בו במיקום הרלוונטי, ואם יש שם אפס, אז נכתוב שם 1 ע"י הוספת החזקה המתאימה של 10.
- על מנת לחשב את הספרה במקום ה-i במספר כלשהו, אנו נשתמש בנוסחה:

$$d_i = \left\lfloor \frac{num}{10^i} \right\rfloor \ \% \ 10$$

סימון הספרות שניחש השחקן

```
for (digit_pos=0, num_tmp = num;
     num_tmp > 0;
     num_tmp /= 10 , ++digit_pos)
{
  current_dig = num_tmp % 10;
  if (current_dig == digit)
  {
    digit_status =
       known_dig  / (int)pow10(digit_pos) % 10 ;
    if (digit_status == 0) {
       known_dig += (int)pow10(digit_pos);
    }
  }
}
```

הדפסת הספרות הידועות

נסרוק במקביל את הספרות של **num** ושל **known_dig**,
ונדפיס כל פעם את הספרה המתאימה, או תו '-', לפי הצורך:

```
num_tmp         = num;
known_dig_tmp   = known_dig;

while (num_tmp > 0)
{
  current_dig   = num_tmp       % 10;
  is_known = known_dig_tmp % 10;
  putchar(is_known ? current_dig+'0' : '-');
  num_tmp       /= 10;
  known_dig_tmp /= 10;
}
```


זיהוי סיום המשחק

המשחק מסתיים רק כאשר לא מודפסים כל תווי '-'. לכן, אנו נניח בהתחלה שהמשחק הסתיים דווקא, ואז תוך כדי הדפסת הספרות הידועות (בשקף הקודם), נוסיף בדיקה כך שאם אנו מדפיסים תו '-' כלשהו – המשחק נימשך !

```
revealed = 1;
while (num_tmp > 0)
{
  …
  is_known = known_dig_tmp % 10;
  if (!is_known)
     revealed = 0;
  putchar(is_known ? current_dig+'0' : '-' ));
  …
}
```

זיהוי סיום המשחק

רק לשם הדגמה, נראה כיצד ניתן היה לקצר את הקוד הקודם בעזרת האופרטור פסיק. בקוד הבא, במידה ואנו מדפיסים את התו '-', המשתנה **revealed** מאופס אוטומטית. קחו בחשבון, שבאופן כללי כתיבה כזו מקשה על קריאת הקוד ואינה מומלצת.

```
revealed = 1;
while (num_tmp > 0)
{
  …
  putchar(is_known ?
     current_dig+'0' : (revealed = 0 , '-') ));
  …
}
```


התוכנית המלאה (1)

הנה התוכנית המלאה. בקוד המלא שלהלן נוסף הטיפול במשתנה **tries_left**, האינדקס שסופר את מספר הניחושים השגויים.

```
#define TRY_NUM 5;

int main()
{
  unsigned int num, known_dig, num_tmp, known_tmp;
  int digit_pos, good_guess, tries_left, revealed, digit;
  int current_dig, digit, digit_status, is_known;

  /* choose random secret number */

  srand(time(0));
  num = rand();
  known_dig = 0;
  …
```

התוכנית המלאה (2)

```
…
revealed = 0; tries_left = TRY_NUM;
while ((tries_left>0) && (!revealed))
{
  /* read a digit from the user */

  do {
    printf("\nenter a digit (tries left: %d): ",
           tries_left);

    if (scanf("%d", &digit) != 1) {
      printf("invalid digit.. bye!");
      return 0;
    }
  } while (digit<0 || digit>9);
  …
```


התוכנית המלאה (3)

```
…
/* locate digit in secret number */

good_guess = 0;
for (digit_pos=0 , num_tmp = num;
     num_tmp>0;
     num_tmp /= 10 , ++dig_pos)
{
  current_dig = num_tmp % 10;
  if (current_dig == digit) {
    good_guess = 1;
    digit_status = known_dig /(int)pow10(digit_pos) % 10 ;
    if (digit_status==0)
      known_dig += (int)pow10(digit_pos);
  }
}
…
```

התוכנית המלאה (4)

```
    …
    if (!good_guess) {
       tries_left--;
       printf("wrong guess!\n");
    }

    /* print the digits known so far */

    num_tmp = num; known_tmp = known_dig; revealed = 1;
    while (num_tmp > 0) {
      digit    = num_tmp   % 10; num_tmp   /= 10;
      is_known = known_tmp % 10; known_tmp /= 10;
      putchar(is_known ? digit+'0' : (revealed=0, '-'));
    }
  }
  return 0;
}
```