# מבוא לתכנות II

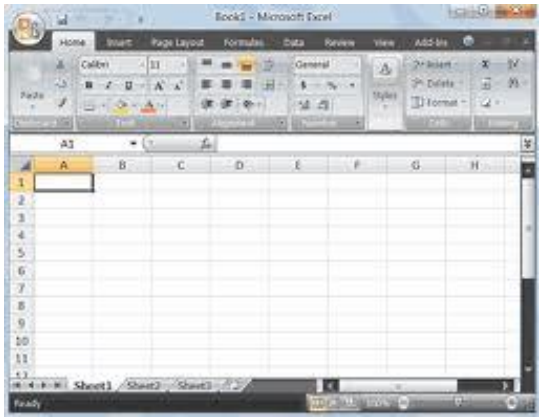
## הרצאה 12 – ייצוג מידע, קבצים ותיקיות

סמסטר 2

# מידע

- מחשבים שומרים ומתפעלים סוגים רבים של מידע:

```
Hello, World!
```



# נתונים

- אבל בשורה תחתונה, המחשב הוא מכשיר אלקטרוני אשר מבין שפה אחת בלבד:

יש חשמל

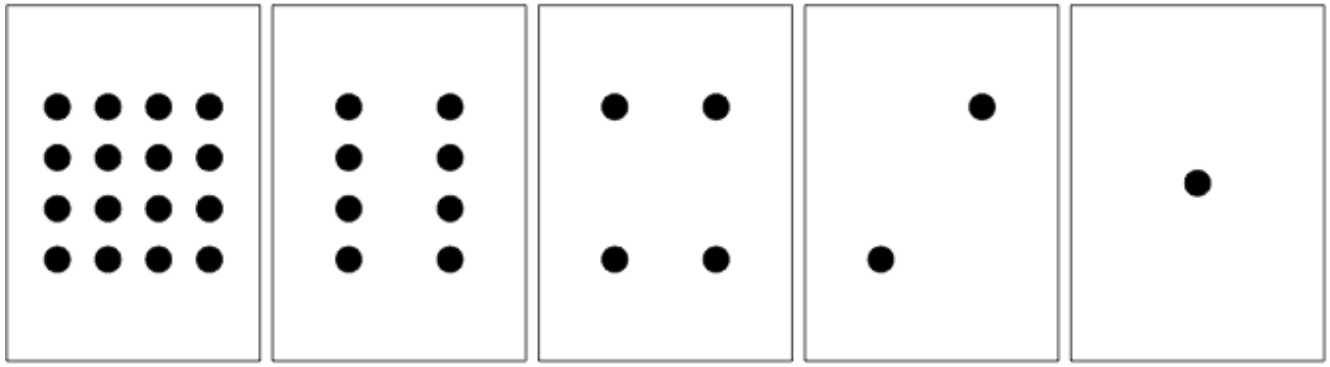
אין חשמל

- אנו קוראים למצב של "אין חשמל" : 0.
ולמצב של "יש חשמל" : 1.

# נתונים

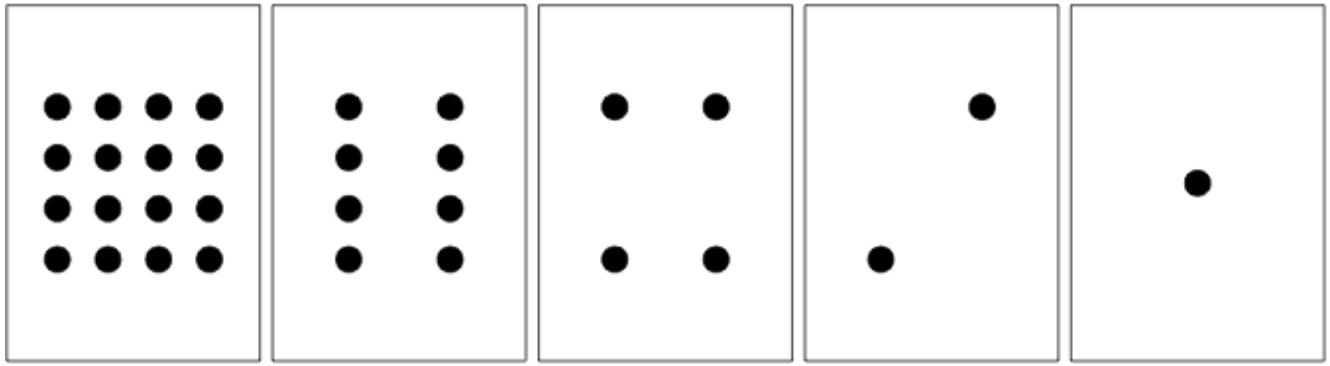
- אז מחשבים מבינים רק 0 ו1.
- אנו צריכים "לתרגם" את המידע ה"אנושי" לשפה הדו-מצבית הזו.
- בואו נתחיל עם הנתונים הקלים ביותר: מספרים.
- איך נייצג מספר חיובי באמצעות 0 ו1?

# מספרים בינאריים



- איך נוכל להיעזר בקלפים הנ"ל כדי לייצג, למשל, את 21?
  - נוכל להשתמש רק בקלף אחד מכל אחד.

# מספרים בינאריים

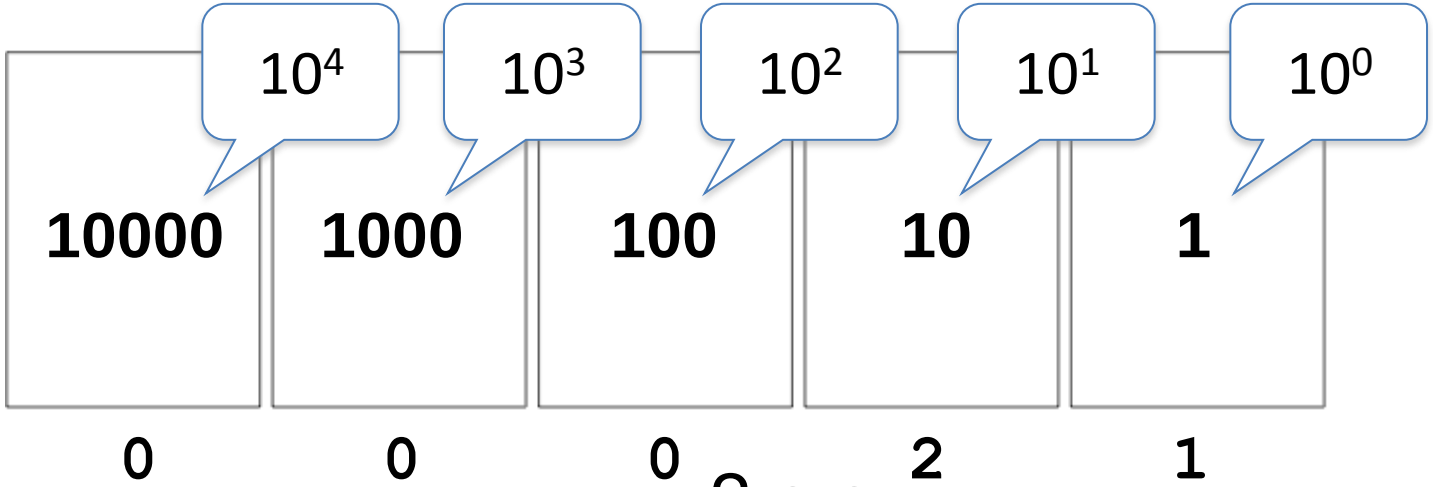


1 0 1 0 1

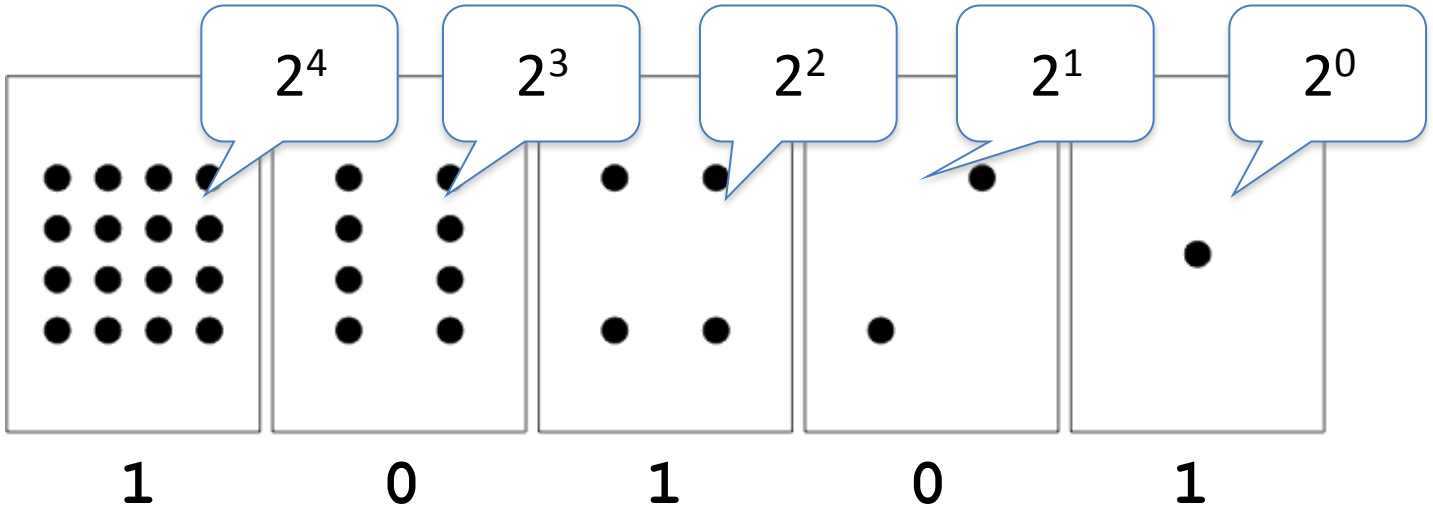
- 21 = 1*16 + 0*8 + 1*4 + 0*2 + 1*1
- נוכל לייצג את המספר 21 באמצעות 0 ו1 בתור המספר 10101.
- נוכל לייצג את המספר 30? כיצד?

# מספרים בינאריים

- אנו רגילים לחשוב בבסיס 10:




- מחשבים חושבים בבסיס 2:



# מספרים בינאריים

- בבסיס 10, אשר גם נקרא <u>בסיס דצימלי</u>: כל מספר הוא סכום של 1-ים, 10-ים, 100-ים וכו'...
  - 21 is 2*10 + 1*1
  - 308 is 3*100 + 0*10 + 8*1

- כאשר משתמשים בבסיס בינארי (בסיס 2), נסכום חזקות של 2 (במקום חזקות של 10): 1, 2, 4, 8, 16...
- זה למה המספר 21 בבסיס 10 הופך למספר 10101 בבסיס 2!

# מספרים בינאריים - תרגיל

אלגוריתם להמרת מספר מבסיס 10 לבסיס 2:

1. `result` = 0
2. `digit` = 0
3. while `number` is positive:
   1. `shift` = $10^{digit}$
   2. Add (`number` % 2) * `shift` to `result`
   3. Divide `number` by 2 (integer division)
   4. Increment `digit`
4. return `result`

כתבו את הפונקציה decimal2binary(number)

```
public int Decimal2binary(int number)
{
    int result = 0, digit = 0;
    while (number > 0)
    {
        int shift = (int)Math.Pow(10, digit);
        result += (number % 2) * shift;
        number /= 2;
        digit ++;
    }
    return result;
}
```

# ייצוג נתונים

- מסקנות עד כה:
  - מחשבים מבינים רק 0-ים ו1-ים.
  - נוכל לייצג מספרים חיוביים באמצעות שימוש ב0-ים ו1-ים בלבד.

- למעשה, נוכל לייצג כל מידע מספרי במחשב – באמצעות מספרים בינאריים!

**1567995227**

1011101011101011011010101011011

# ייצוג טקסט

- זכרו: תו מייצג אות, ספרה או סמל:

| A | B | C | a | b | c | % | ? | & |
|---|---|---|---|---|---|---|---|---|
| 65 | 66 | 67 | 97 | 98 | 99 | 37 | 63 | 38 |

- לכל תו יש מספר זהות. מספר זה מאושר ומוכר על ידי כל המחשבים בעולם!
- כשאנו מקלידים תווים, הם למעשה מיוצגים בתור מספרים.
  - ואנו יודעים כיצד לייצג מספרים...

# ביטים (Bit) ובייטים (Bytes)

- ספרה בינארית (binary digit) נקראת ביט (bit).
- רצף של 8 ביטים נקרא בייט.
- 1024 בייטים נקראים קילו-בייט (KB)
- 1024 KB נקראים מגה-בייט (MB)
- 1024 MB נקראים Giga-Byte (GB)
- וכך הלאה....

# ביטים ובייטים

דוגמה בשיווק: ספקי אינטרנט


מה הכוונה במהירות 100 מגה?

איזו מהירות הדפדפן יציג בפועל?

מדוע יש הבדל?

# ייצוג טקסט

ASCII:

- בהתחלה, הייצוג של תווי טקסט עשה שימוש בבייט אחד.
  - כמה תווים שונים ניתן לייצג בדרך זו?
  - אתם רואים את הבעיה בגישה זו?

Unicode:

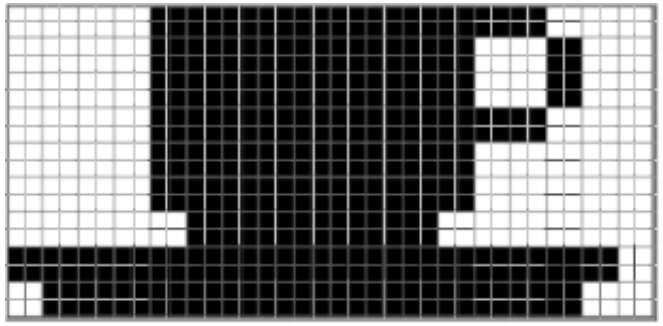
- אנו רוצים שמחשבים בכל העולם יוכלו להציג טקסט בשפות רבות.
- 256 תווים לא מספיקים עבור כל שפות העולם!
- Unicode משתמש ב 2 בייטים עבור כל תו:
  - 65.536 תווים שונים!

# ייצוג תמונות

- מה בנוגע לתמונות?
- איך נמיר מידע כה מורכב ל0-ים ו1-ים?

- נוכל לחלק את התמונה לרשת של נקודות קטנות, כל נקודה בעלת צבע משלה.
- במונחי מחשוב, כל נקודה שמה: פיקסל.
- אתגר חדש: נחליט כיצד לייצג צבעים.

# ייצוג תמונות

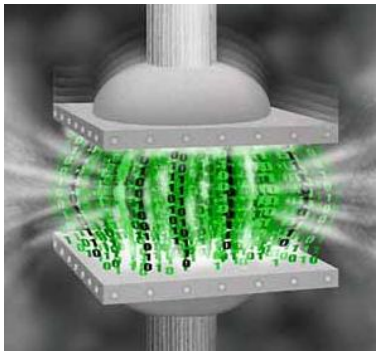


- בואו נתחיל עם תמונות בשחור-לבן.
- גודל תמונה זו 36X18 פיקסלים.
- ייצוג פשוט שלה הינו ייצוג הפיקסלים השחורים בתור 1-ים, והלבנים בתור 0-ים.
- זה ידרוש ביטים 36*18 = 648 אשר הם 81 בייטים.
- נוכל, במקום זאת, לשמור את האורכים של הרצפים בעלי אותו הצבע:
  - לדוגמה, בשורה הראשונה יש 8 פיקסלים לבנים, 22 שחורים ואז עוד 6 פיקסלים לבנים.
  - נוכל לייצג שורה זו בתור: 8,22,6 (כל מספר הוא בייט)
  - כל התמונה תיקח רק 60 בייטים – 26% פחות!

# ייצוג תמונות

רעיונות נוספים להקטנת הגודל:

- השמטת אורך הרצף האחרון וחישובו, בידיעת אורך השורה.
- ספירה של זה בתור שורה אחת ארוכה ואז סידורה מחדש כטבלה. זה יוריד את הפיקסלים הלבנים (או השחורים) הראשונים והאחרונים למספר אחד במקום שניים.
- הורדת גודל המספר שיש לשמור ל5 ביטים במקום בייט, בגלל שהשורה קטנה מ63.
- או...

# דחיסת מידע

- צמצמנו את גודל התמונה ב26%.
- אבל מדוע שנעשה זאת?
  - מצלמות דיגיטליות: תמונות קטנות יותר מובילות לצורך מוקטן של שטח אחסון ולכרטיסי זיכרון זולים יותר.


  - שרתי אינטרנט (חישבו על פייסבוק או פיקאסה): תמונות קטנות יותר מובילות לצמצום שרתים, פחות דיסקים.
  - תנועה באינטרנט: מיליוני תמונות נשלחות ומורדות בכל יום. כל ביט שנצמצם משמעותו אינטרנט מהיר עבור כולנו!

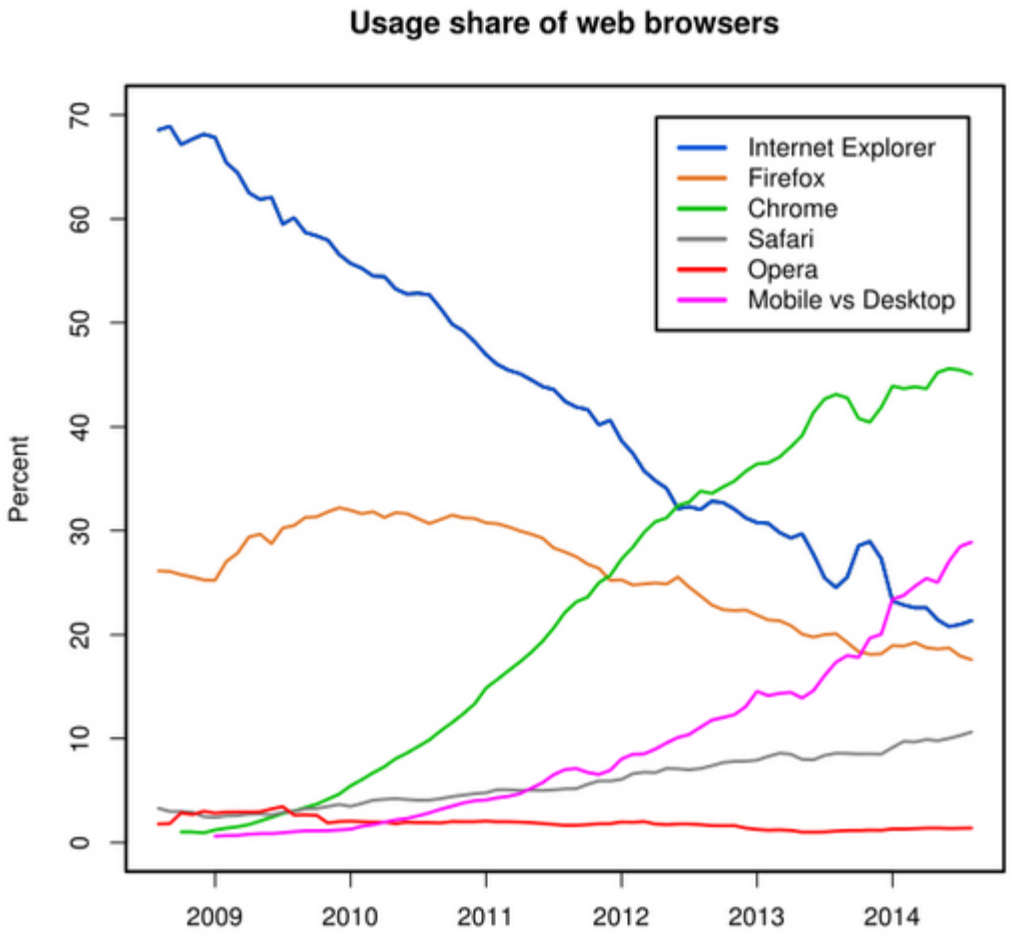
facebook Picasa flickr

# דחיסת מידע

- כיום, נתונים נדחסים בכל תחום:
  - תמונות (.jpg, .gif, ...)
  - סרטים (DVD, downloads, you-tube, ...)
  - קבצי קול (MP3)
  - מסמכים (Zip)
  - דפי אתרים (Zip, Google Chrome)


- דחיסת נתונים היא תחום מחקר פרקטי למדי – הפחתת זמן / מקום מעידים על חסכונות גדולים.

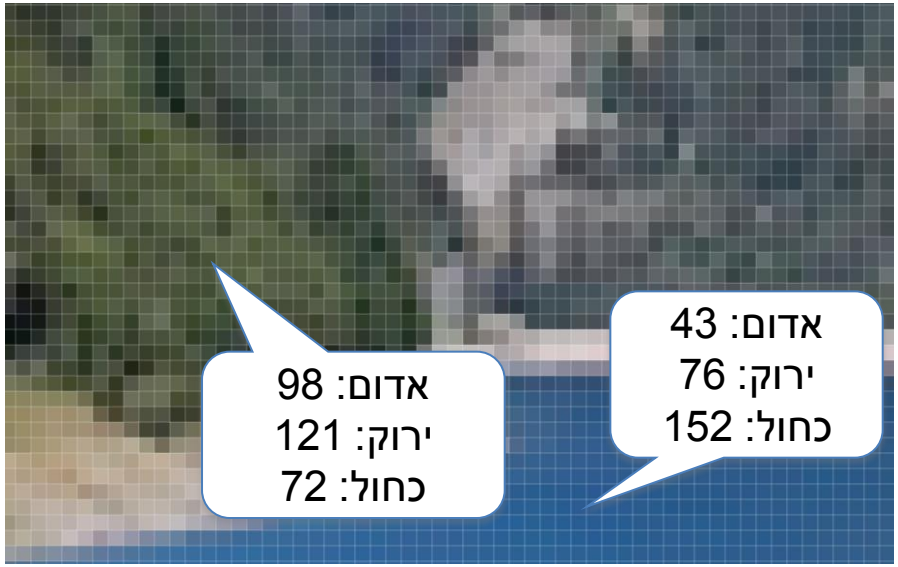
# דחיסת מידע


Usage share of web browsers
Percent
0
10
20
30
40
50
60
70
2009
2010
2011
2012
2013
2014
Internet Explorer
Firefox
Chrome
Safari
Opera
Mobile vs Desktop

# ייצוג תמונות צבעוניות


- נוכל לייצג כל צבע באמצעות שימוש בצבעי היסוד: אדום, ירוק וכחול.
- ערבובים שונים של צבעי היסוד יספקו לנו צבעים חדשים.
- עבור כל פיקסל, נחליט על *הפרופורציות* של כל צבע בערבוב
  - *פרופורציות* הן מספרים. אנו יודעים כיצד לייצג מספרים.
  - משתמשים ב3 מספרים שונים כדי לייצג את הצבע של כל פיקסל.
- כל מה שנותר לנו לעשות הוא לייצג את כל הפיקסלים של התמונה...

# ייצוג תמונות צבעוניות




- אנו משתמשים במספרים בטווח של 255 – 0 כדי לייצג כל צבע (אדום/ירוק/כחול). מדוע?
- כל פיקסל מיוצג תוך שימוש ב3 בייטים (=24 ביט)
  - תמונות בצבע גדולות פי 24 מאשר תמונות בשחור/לבן!
- ישנן טכניקות דחיסה עבור תמונות בצבע
  - BMP
  - JPEG
  - GIF
  - PNG
  - ועוד...

# ייצוג סרטים

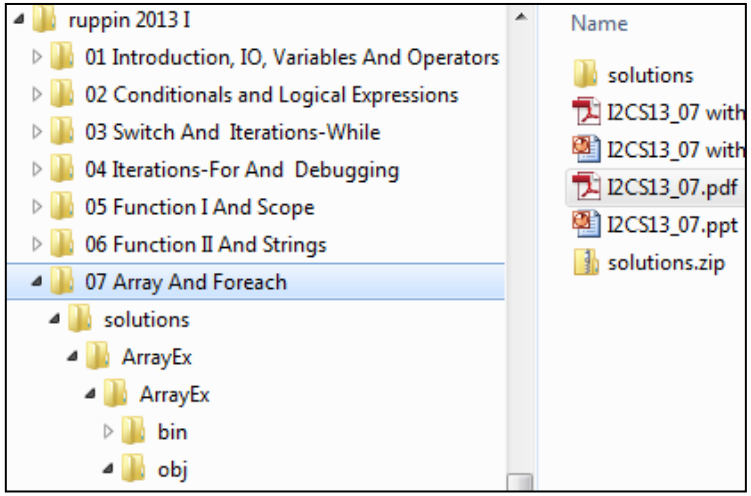
- סרטים מכילים:
  - כ30~ תמונות צבעוניות כל שניה.
  - ערוצי קול.
- נוכל לייצג סרט בתור סידרה של תמונות מרובות, בתוספת של ערוצי קול.
- אך, כמות התמונות הינה עצומה!
  - סרט באורך 10 שניות הוא כ300 תמונות.
  - סרט באורך שעה הוא כ108,000 תמונות!
- אתם יכולים לחשוב על דרך לדחוס נתון תמונה בסרט?

# קבצים

- קובץ הוא רצף של בייטים אשר מייצגים מידע כלשהו.
- קובץ יכול לייצג טקסט, תמונה, שיר, קוד C# ועוד...
- ניתן לאחסן קובץ בדיסק הקשיח של המחשב
  - נוכל לטעון אותו שוב אחר כך.
  - או לשלוח אותו דרך האינטרנט.
  - או לשמור אותו במקום אחסון נייד כמו USB / CD.
  - או להעלות אותו לאתר הקורס.

# מערכת קבצים

- קבצים בדרך כלל מאורגנים במבנה היררכי של תיקיות.
  - אנו קוראים למבנה זה "עץ התיקיות".
- כדי לגשת לקובץ עלינו לדעת:
  - את שמו.
  - את המיקום שלו.
- מיקום הקובץ מתואר בתור המסלול בעץ התיקיות:
  - In Windows: C:\Users\me\My Documents\myfile.txt
  - In Unix: /Users/me/Documents/myfile.txt



# פתיחת קובץ

- כדי לגשת לקובץ, נפתח אותו קודם.
  - הכוונה לכך שנבקש ממערכת ההפעלה רשות לגשת לקובץ.
- לאחר מכן, יהיה ברשותנו אובייקט קובץ (*file object*) שנוכל לעבוד עמו.
- נוכל לפתוח קובץ עבור קריאה ועבור כתיבה.
- ב#C, קיימות כבר 2 מחלקות שמטפלות בפתיחת הקובץ עבורנו: StreamReader ו- StreamWriter.
- הן דורשות הוספה של: System.IO;

# קריאת קובץ

- נניח כי יש לנו קובץ בשם `file.txt`, אשר מאוחסן ב `C:\temp`
- תוכן הקובץ הזה הוא:

```
This is a file
with some data in it
```

- נרצה לקרוא תוכן זה:

```
StreamReader sr = new
StreamReader(@"c:\temp\file.txt");
string firstline = sr.ReadLine();
string strInput = sr.ReadToEnd();
sr.Close();
```

צור זרם אל 'C:\temp\file.txt' עבור קריאה. @ - התעלם מתווים מיוחדים

קורא את השורה הבאה בתור מחרוזת אחת

קורא את שאר הקובץ בתור מחרוזת אחת

סוגר ומשחרר את הקובץ **(חשוב מאוד!!)**

# קריאת קובץ

- `file.read()`
  קורא תו אחד בתור מספר.
- `file.Peek()`
  קורא ומחזיר את התו הבא אחר לא מזיז את הסמן. מחזיר 1- אם לא נותר מה לקרוא.
- `StremReader.EndOfStream`
  מחזיר בוליאני אם הגענו לסוף הקובץ.

# קריאת קובץ

- נוכל להיעזר בלולאת "for" כדי לרוץ על כל השורות בקובץ:

```
StreamReader sr = new StreamReader(@"c:\temp\file.txt");
while (!sr.EndOfStream)
{
    label3.Text += sr.ReadLine().TrimEnd() + "\n";
}

sr.Close(); //if we comment this, after running this function
            //we could not change the file.txt
            //from windows because it is still used by this
process
```

מעלים רווחים מיותרים ויורד שורה בסוף השורה

- כאשר קוראים קבצים גדולים, זה לעתים טוב יותר לקרוא את הקובץ שורה-שורה, ולא הכל בבת אחת. (זיכרון מול זמן)

# כתיבת קובץ

- נניח כי יש לנו קובץ בשם mydata.txt, אשר מאוחסן ב C:\temp
- התוכן שנרצה לכתוב בקובץ הוא:

```
This is my data
that i need to store
```

- נרצה לשמור את התוכן זה:

```
string[] arr = new string[] {"This is my data",
                             "that i need to store" };

StreamWriter sw = new StreamWriter(@"c:\temp\mydata.txt");
//will create the file if not existed

for (int i = 0; i < arr.Length;
{
    sw.Write(arr[i] + "\r\n");
    //sw.WriteLine(arr[i]); //the same as above
}
sw.Close();
```

כותב את המחרוזת אל תוך הקובץ. ירידת שורה באמצעות r/n/

צור זרם אל 'C:\temp\mydata.txt' עבור כתיבה. @ - התעלם מתווים מיוחדים

סוגר ומשחרר את הקובץ **(חשוב מאוד!!)**

# קובץ

- File - למחלקה זו מספר פונקציות לטיפול בקבצים:
  - File.Exists(filePath) – boolean, if exists
  - File.Delete(filePath)
  - StreamWriter sw = File.CreateText( "myFile.txt" )
  - File.CreateText
    - מחזיר sw אז אם לא נאחסן אותו ונסגור, נקבל שגיאה כאשר ננסה למחוק את הקובץ או התיקייה!
    - sw.Close()
  - File.AppendText( filePath ) - הוספה
  - File.AppendAllText("file1.txt", "text to append"); //very useful
  - string[] lines = File.ReadAllLines("file1.txt"); //very useful
  - string str = File.ReadAllText("file1.txt"); //very useful
  - File.OpenText( "myFile.txt" ) – עבור קריאה
  - File.Move(srcfilePath, destfilePath) – שינוי מיקום
- filePath– יחסי לקובץ הexe.
- "\r\n" – שורה חדשה בקבצים.

# תיקייה

- תיקייה
  - Directory.Exists(*directoryPath*)
  - Directory.CreateDirectory(*path*)
  - Directory.Delete(*path*,recursive=t*rue/false) – recursive= מחיקת גם תיקיות משנה וקבצים*
  - Directory.Move*(source path, destination path)*
  - Directory.GetDirectories(*path*) – מחזיר מערך מחרוזות, שמות תתי-התיקיות
  - Directory.GetFiles(*path* [, *search pattern*]) – מחזיר מערך מחרוזות, שמות הקבצים
  - Directory.GetFileSystemEntries(*path*) – קבצים ותיקיות
  - Directory.GetCurrentDirectory();

** directoryPath - יחסי לקובץ ה-exe*

# תוספות

- "\r\n" – שורה חדשה.

  זכרו – זה תופס 2 תווים/מקומות!
- נסו לעבוד עם זרם פתוח אחד, כדי להימנע מפקודות פתיחת/סגירת קבצים שגוזלות זמן.
- כל מחלקה תשמור/תטען את עצמה.
- openFileDialog – כדי לפתוח קובץ מקומי.
  - openFileDialog1.ShowDialog();
- FileInfo – כדי לקבל מידע על הקובץ
  - FileInfo fi = new FileInfo(fullName);
  - Name
  - Extension
  - CreationTime
  - DirectoryName

# תרגיל

| Name | Date modified | Type | Size |
|---|---|---|---|
| Files2.exe | 21/10/2012 16:04 | Application | 12 |
| Files2.pdb | 21/10/2012 16:04 | PDB File | 32 |
| Files2.vshost.exe | 02/05/2011 15:22 | Application | 12 |
| Files2.vshost.exe.manifest | 17/03/2010 21:39 | MANIFEST File | 1 |
| nir.txt | 21/10/2012 16:06 | Text Document | 1 |
| notes.txt | 21/10/2012 16:08 | Text Document | 1 |