

מבוא לתכנות II

הרצאה 12 – ייצוג מידע, קבצים ותיקיות

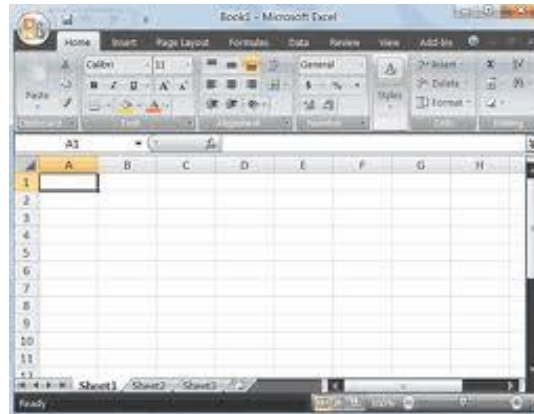
סמסטר 2

מידע

- מחשבים שומרים ומתפעלים סוגים רבים של מידע:

Hello, World!

1 2 3

A screenshot of the Facebook sign-up page. It includes the Facebook logo, login fields for email and password, and a sign-up section with fields for first name, last name, email, and password, along with a "Sign Up" button.

נתונים

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



אין חשמל



יש חשמל

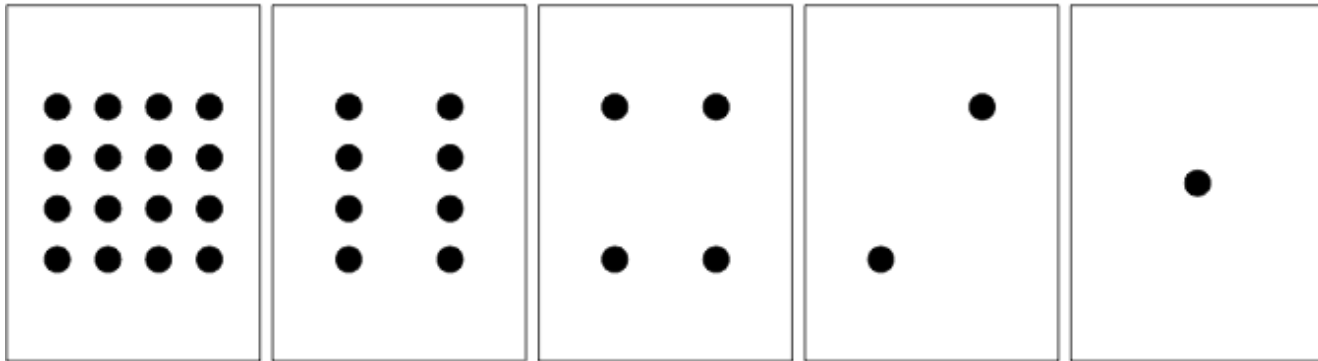
- אנו קוראים למצב של "אין חשמל" : 0.
ולמצב של "יש חשמל" : 1.

נתונים

- אז מחשבים מבינים רק 0 ו 1.
- אנו צריכים "לתרגם" את המידע ה"אנושי" לשפה הדו-מצבית הזו.
- בואו נתחיל עם הנתונים הקלים ביותר: מספרים.
- איך נייצג מספר חיובי באמצעות 0 ו 1?

1²₃

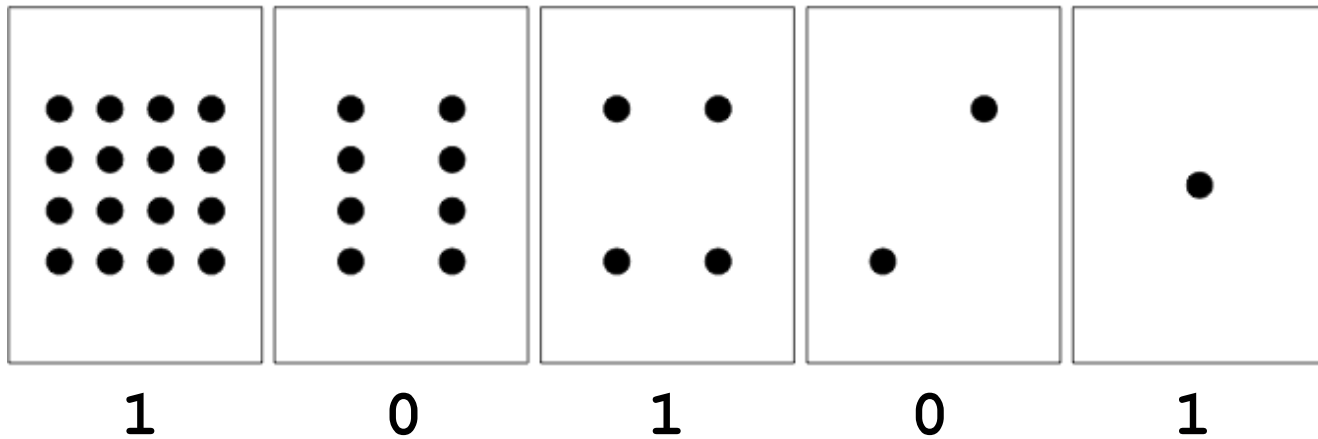
מספרים בינאריים



• איך נוכל להיעזר בקלפים הנ"ל כדי לייצג, למשל, את 21?

– נוכל להשתמש רק בקלף אחד מכל אחד.

מספרים בינאריים



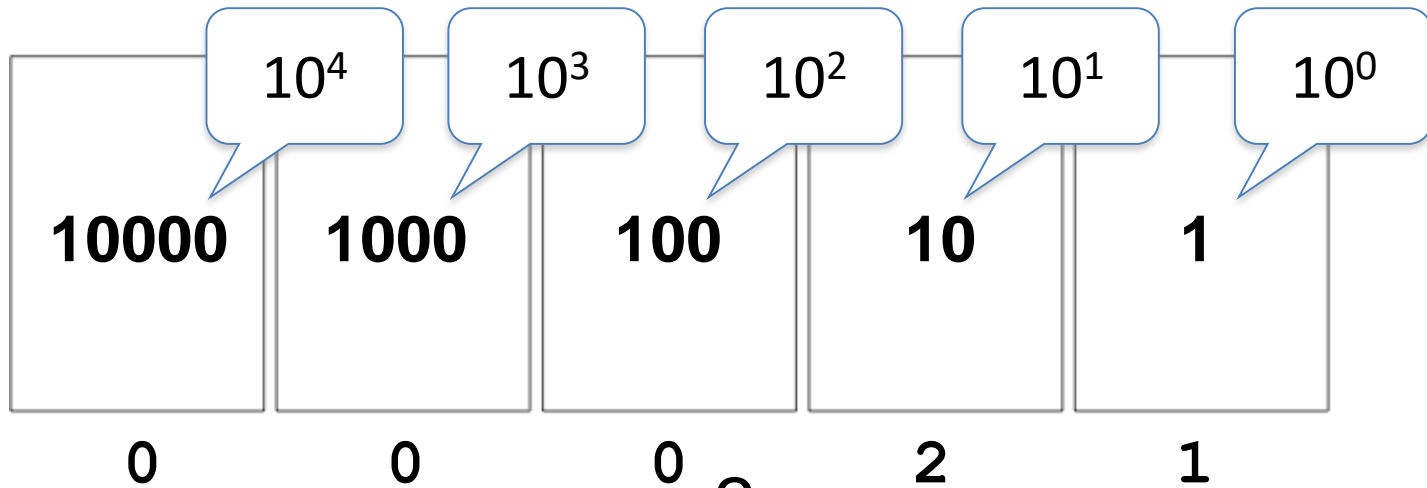
- $21 = 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$

- נוכל לייצג את המספר 21 באמצעות 0 ו 1 בתור המספר 10101.

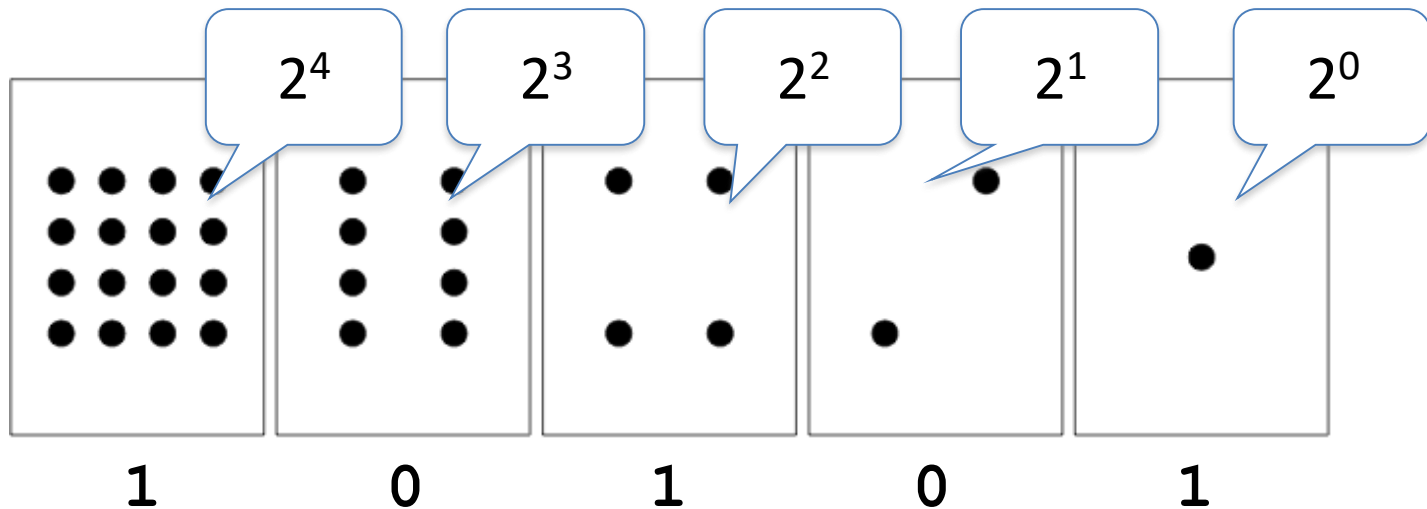
- נוכל לייצג את המספר 30 כיצד?

מספרים בינאריים

- אנו רגילים לחשוב בבסיס 10:



- מחשבים חושבים בבסיס 2:



מספרים בינאריים

- בבסיס 10, אשר גם נקרא בסיס דצימלי: 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 = 10digit`
 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;
}
```

- 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

בחצי מחיר!							
100 מגה	15 מגה	12 מגה	10 מגה	5 מגה	4 מגה	2.5 מגה	1.5 מגה
1/2-ב מחיר!	1/2-ב מחיר!	1/2-ב מחיר!	1/2-ב מחיר!	1/2-ב מחיר!	1/2-ב מחיר!	1/2-ב מחיר!	1/2-ב מחיר!

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

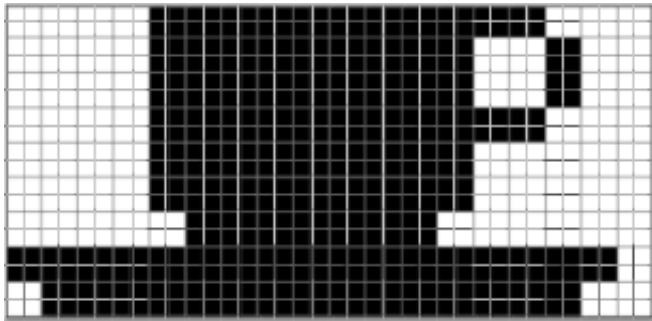
- אנו רוצים שמחשבים בכל העולם יוכלו להציג טקסט בשפות רבות.
- 256 תווים לא מספיקים עבור כל שפות העולם!
- Unicode משתמש ב 2 בייטים עבור כל תו:
 - 65.536 תווים שונים!

ייצוג תמונות



- מה בנוגע לתמונות?
- איך נמיר מידע כה מורכב ל-0 ימים ו-1 ימים?
- 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

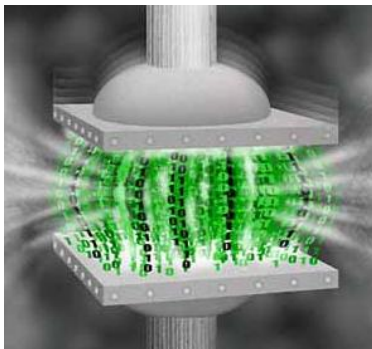


- בואו נתחיל עם תמונות בשחור-לבן.
- גודל תמונה זו 36X18 פיקסלים.
- ייצוג פשוט שלה הינו ייצוג הפיקסלים השחורים בתור 1-ים, והלבנים בתור 0-ים.
- זה ידרוש ביטים $18 \times 36 = 648$ אשר הם 81 בייטים.
- נוכל, במקום זאת, לשמור את האורכים של הרצפים בעלי אותו הצבע:
 - לדוגמה, בשורה הראשונה יש 8 פיקסלים לבנים, 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 ב-26%.

- 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!

The Facebook logo, consisting of the word 'facebook' in white lowercase letters on a blue rectangular background.The Flickr logo, with the word 'flickr' in a blue sans-serif font, where the 'i' is pink and the 'r' is blue.

דחיסת מידע

- כיום, נתונים נדחסים בכל תחום:

– תמונות (.jpg, .gif, ...)

– סרטים (DVD, downloads, you-tube, ...)

– קבצי קול (MP3)

– מסמכים (Zip)

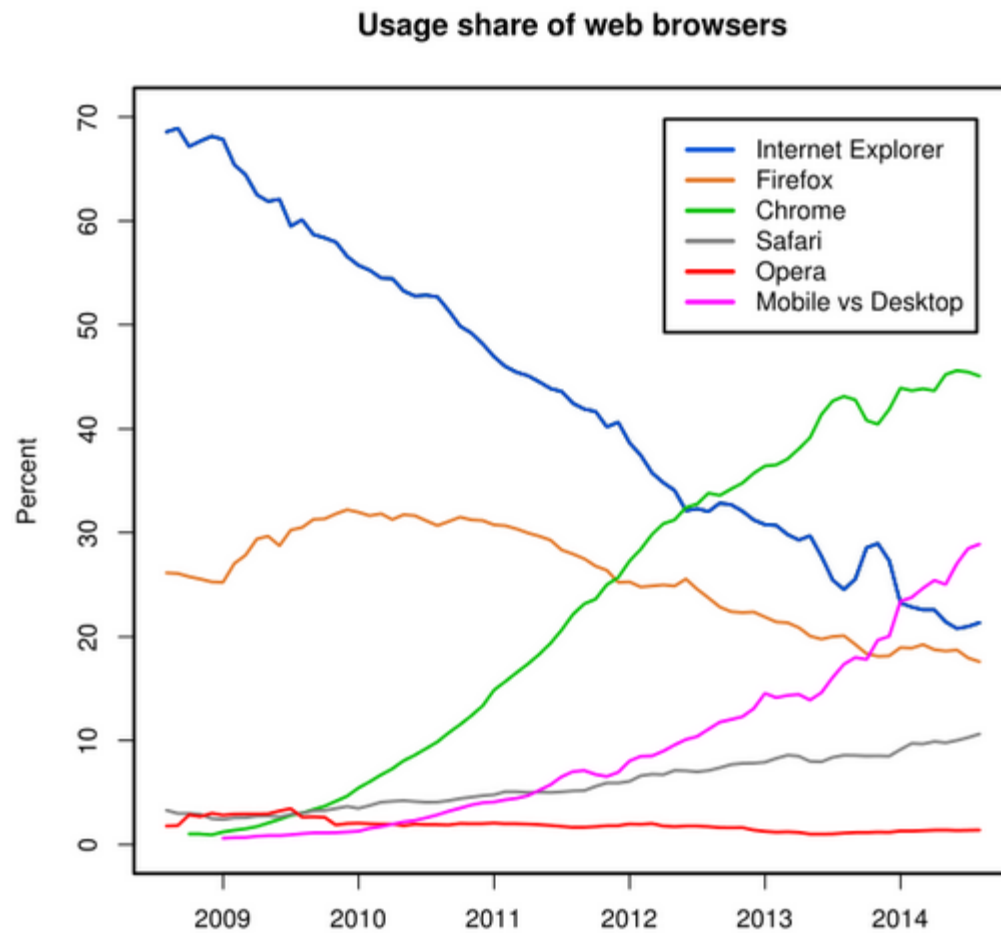
– דפי אתרים (Zip, Google Chrome)



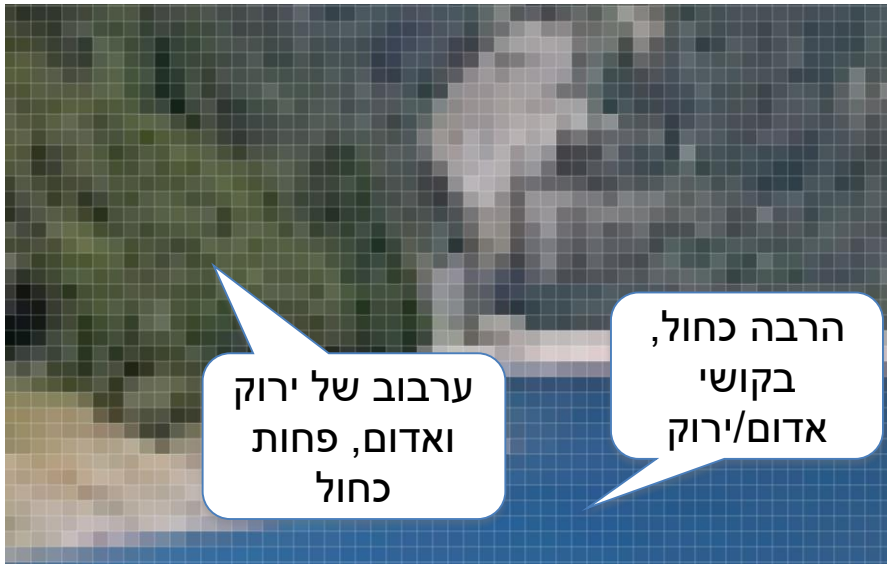
Google Chrome

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

דחיסת מידע

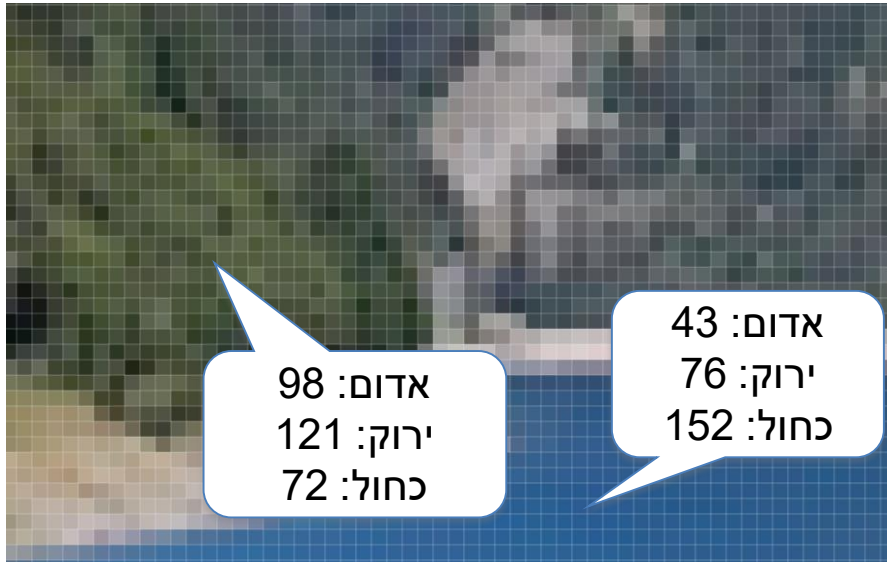


ייצוג תמונות צבעוניות



- 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 ב-3 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 0 – 255 כדי לייצג כל צבע (אדום/ירוק/כחול). מדוע?
- כל פיקסל מיוצג תוך שימוש ב-3 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:
 - כ-30 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 כ-300 תמונות.
 - סרט באורך שעה הוא כ-108,000 תמונות!
- אתם יכולים לחשוב על דרך לדחוס נתון תמונה בסרט?

- 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.

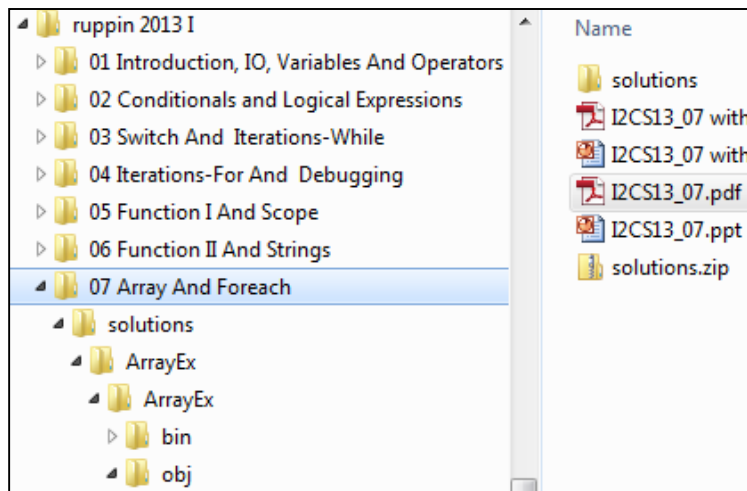
- מיקום הקובץ מתואר בתור המסלול בעץ התיקיות:

– 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`

This is a file
with some data in it

- תוכן הקובץ הזה הוא:

- 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.
- `StreamReader.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`

This is my data
that i need to store

- התוכן שנרצה לכתוב בקובץ הוא:

- 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!!)

- 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 התיקיה!
 - 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 exe.
- "\r\n" – שורה חדשה בקבצים.

- Directory.Exists(*directoryPath*)
- Directory.CreateDirectory(*path*)
- Directory.Delete(*path*, recursive=true/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

