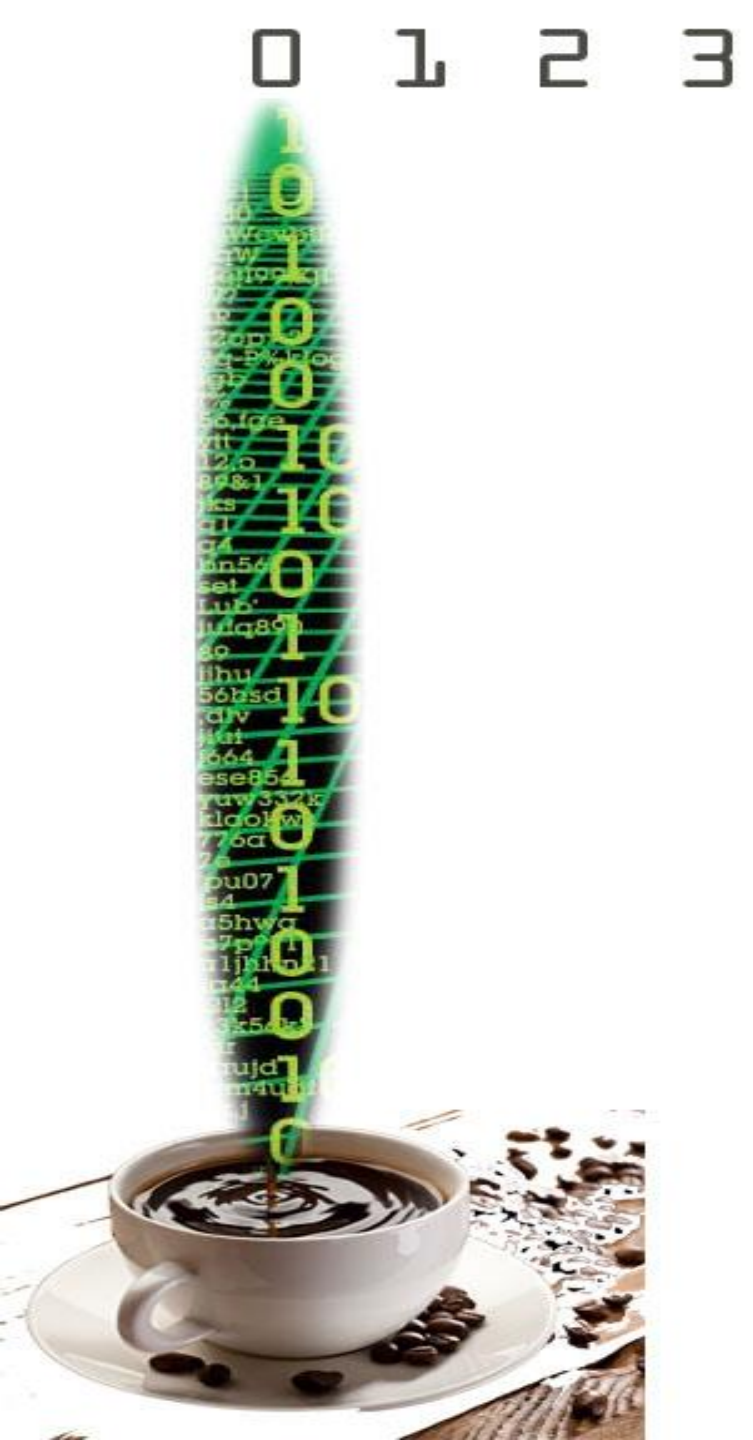


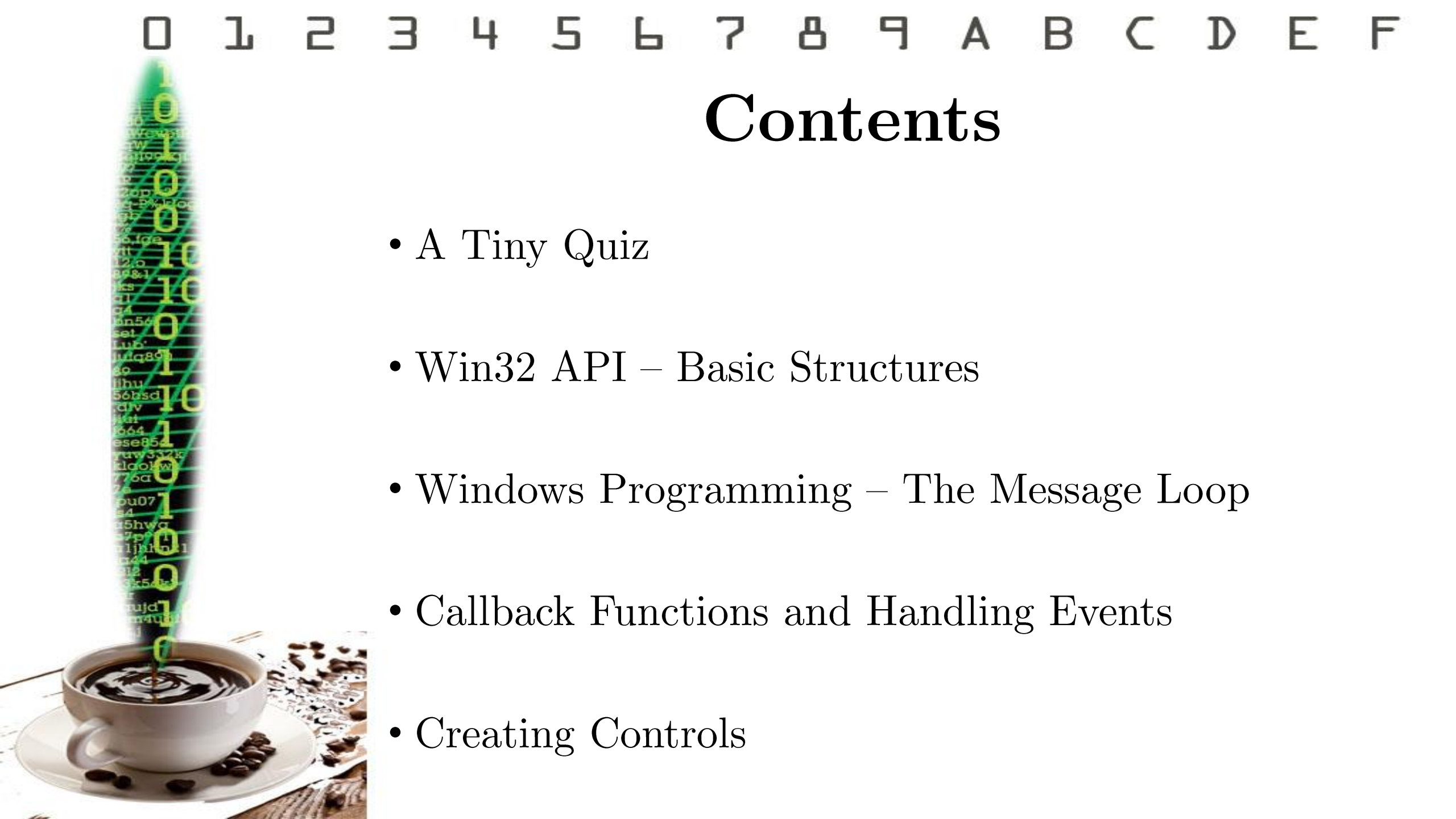
Windows Programming

Building Windows Programs with the Win32 API in C++

Dimitrios N. Terzopoulos

terzopod@math.auth.gr

18/11/2015



Contents

- A Tiny Quiz
- Win32 API – Basic Structures
- Windows Programming – The Message Loop
- Callback Functions and Handling Events
- Creating Controls

Practice in C++

- Questions, anyone??

- Creating a simple window with some Controls.

100$ Question

```
int v() {
    int m=0;
    return m++;
}
```

```
int main() {
    cout<<v();
}
```

Outputs...

1	Compile Error
0	Undefined

100$ Question

```
int v() {
    int m=0;
    return m++;
}
```

```
int main() {
    cout<<v();
}
```

Outputs...

- 1
- Compile Error
- 0
- Undefined

200$ Question

```
int mat[10];
for (int i = 0; i < 10; i++)
        mat[i] = 2*i;
int k = 3;
cout << k[mat] << endl;
```

Outputs...

Error	13
3	6

200$ Question

```
int mat[10];
for (int i = 0; i < 10; i++)
        mat[i] = 2*i;
int k = 3;
cout << k[mat] << endl;
```

Outputs...

- Error
- 13
- 3
- 6

0 1 2 3 4 5 6 7 8 9 A B C D E F

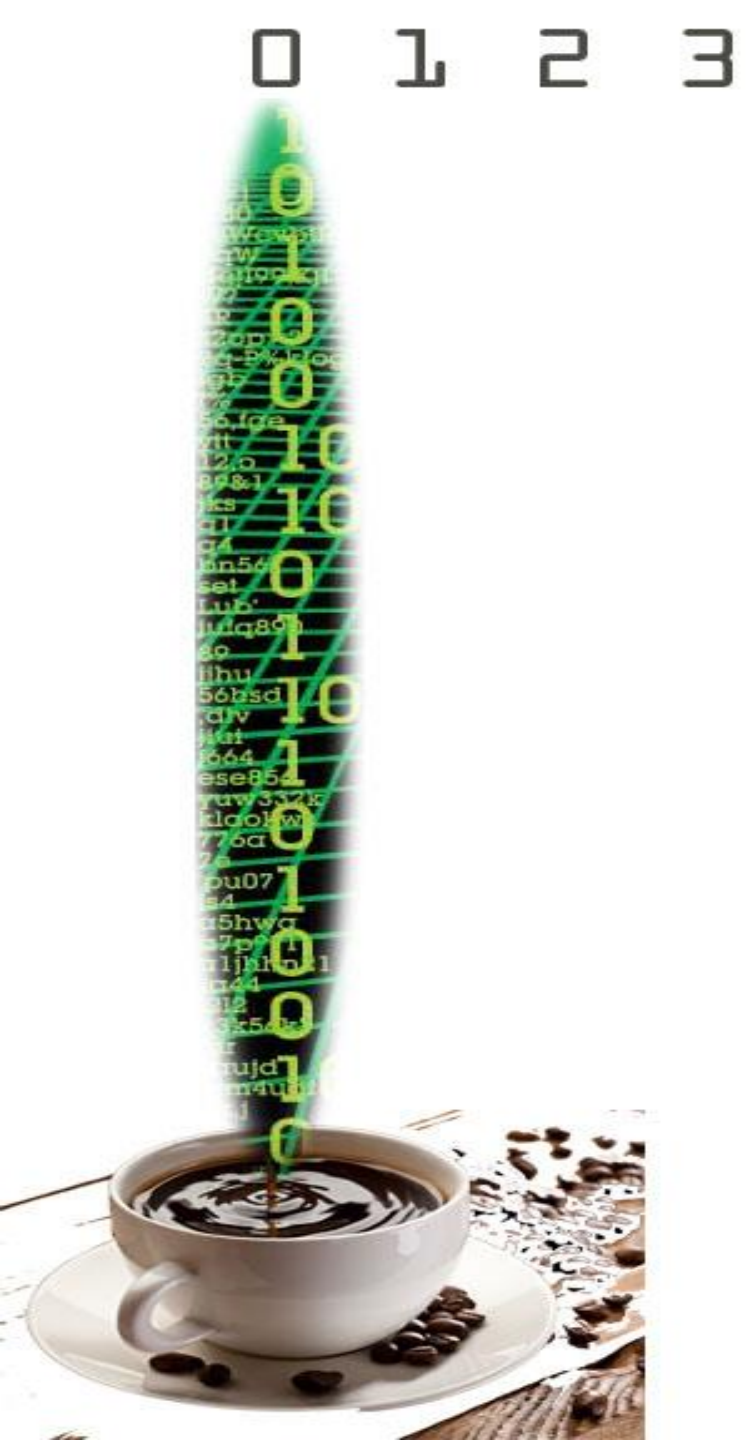
200$ Question

In C++, Matrix[ind] is interpreted as *(Matrix+ind)... a pointer! So...

$$k[mat] \Leftrightarrow *(k+mat) \Leftrightarrow *(mat+k) \Leftrightarrow mat[k]$$

(Don't say it...)

Error	Undefined
3	6



500$ Question

```
short var = 0;
for (int i = 0; i < 60000; i++)
        cout << var++ << endl;
```

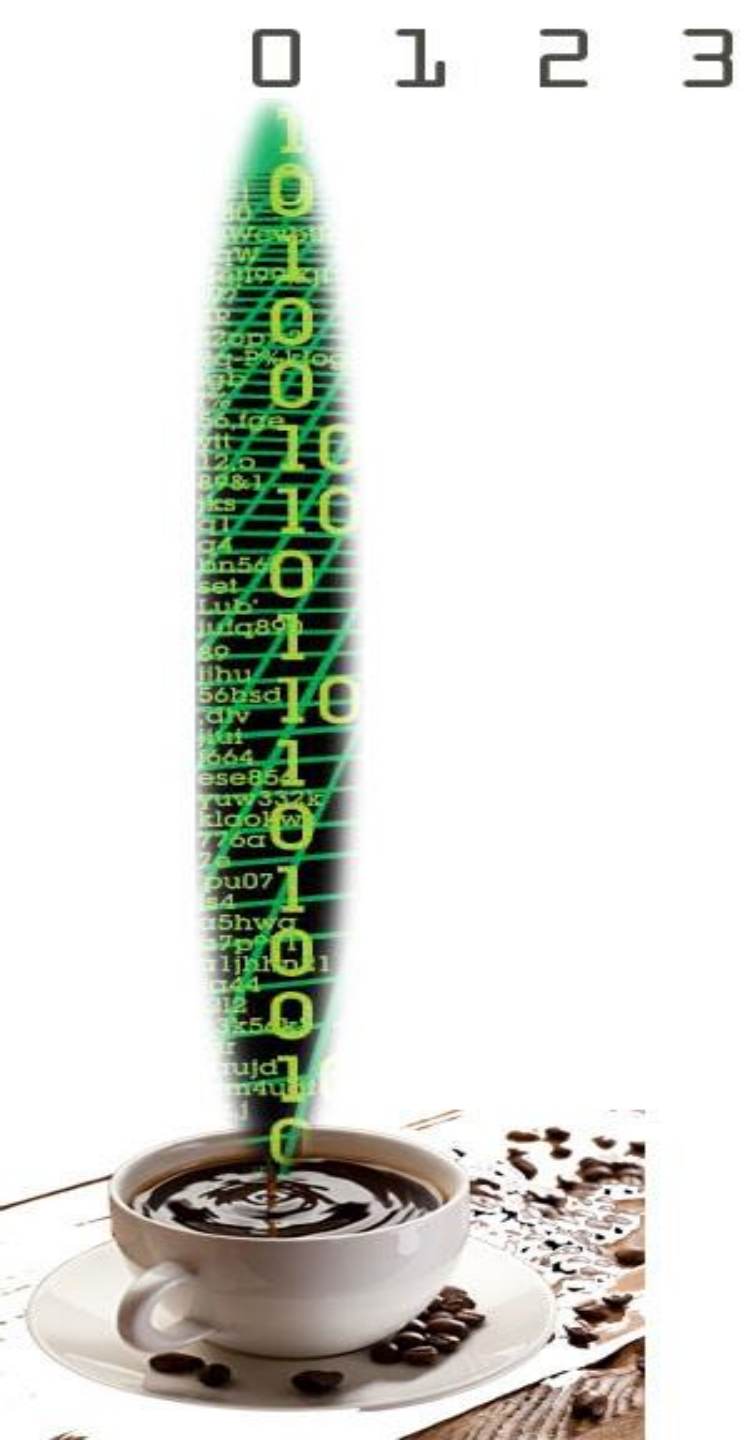
The printed
values will be...

All Positive

All Negative

Positive & Negative

An Error will occur



500$ Question

```
short var = 0;
for (int i = 0; i < 60000; i++)
        cout << var++ << endl;
```

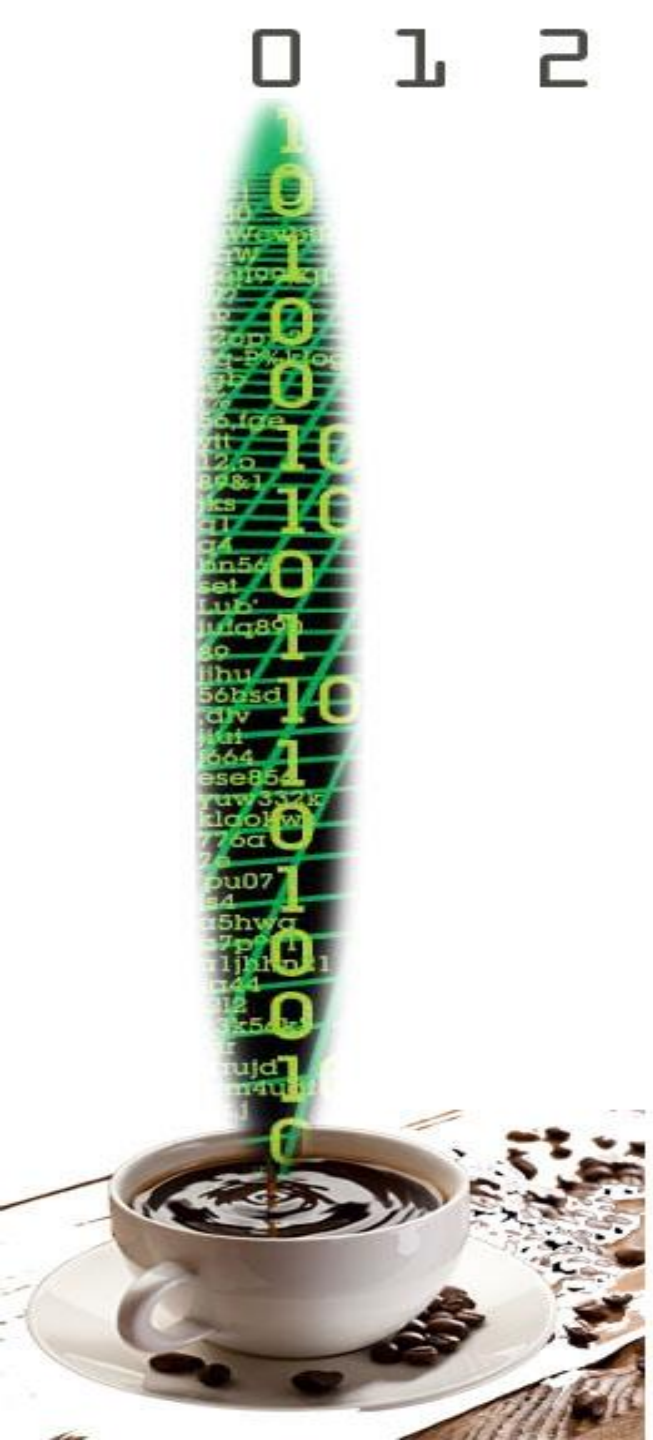
The printed
values will be...

All Positive

All Negative

Positive & Negative

An Error will occur



500$ Question

A short is (usually!) 2 Bytes → 16 bits.

A short, unless otherwise *declared*, is **SIGNED**

From (-2^{15}) to $(2^{15} - 1)$

0, 1, 2, ... 32766, 32767, (*1st bit now set*) -32768, -32767..

All Positive

All Negative

Positive & Negative

An Error will occur

1000$ Question

Which of the following data structures caters for fastest data retrieval?

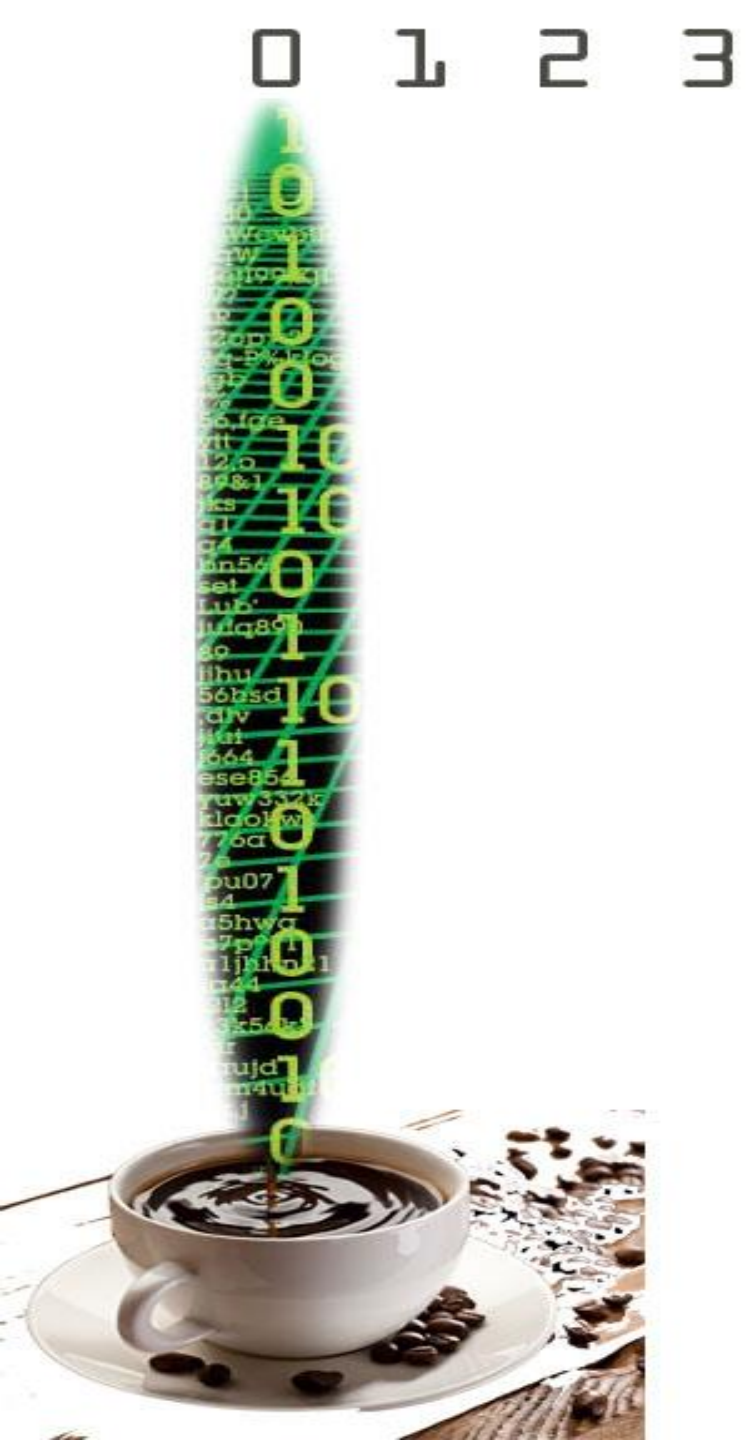
(On Average)

Heap

Hash Table

Stack

Binary Tree



1000$ Question

Which of the following data structures caters for fastest data retrieval?

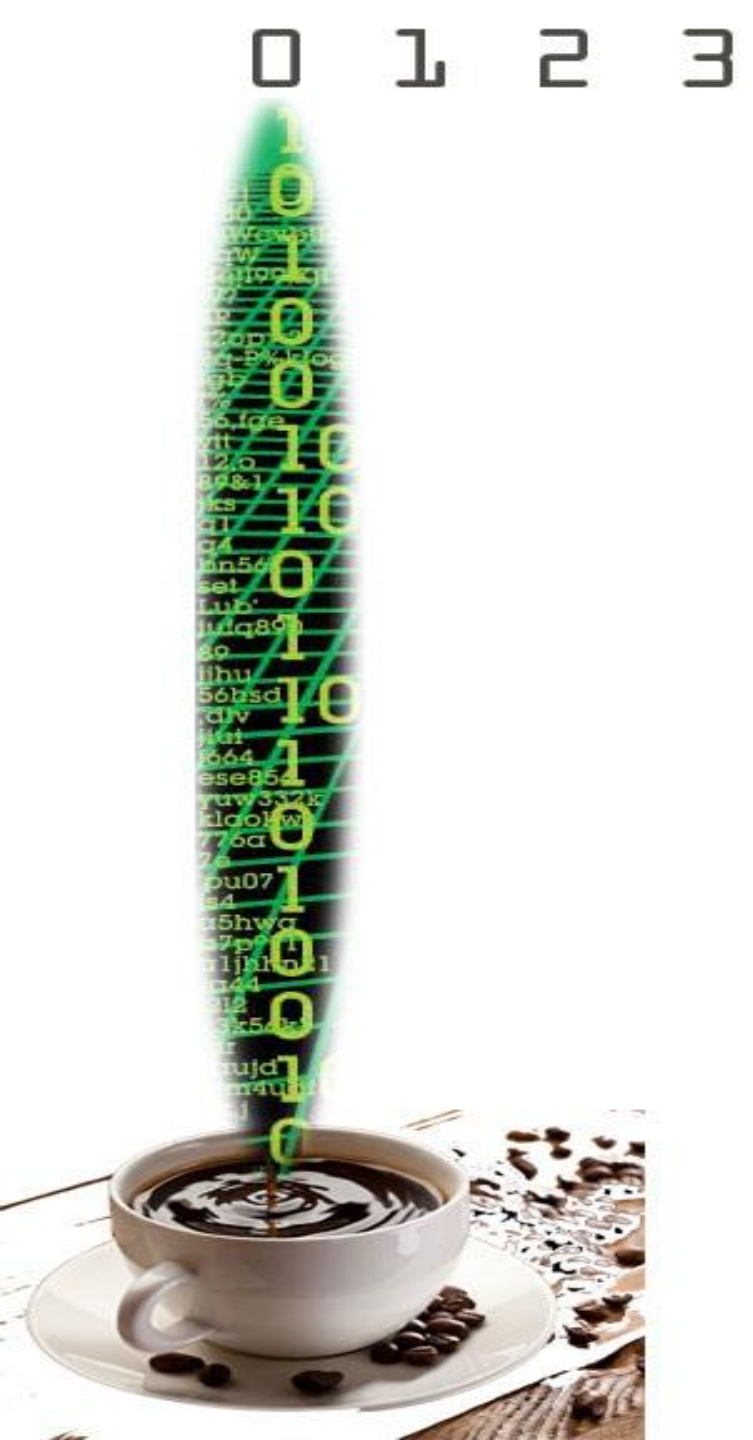
(On Average)

Heap

Hash Table

Stack

Binary Tree



1000$ Question

Any Doubts???

A hash table encodes keys and stores (key,value) pairs. Well, roughly...

Access Complexity:
O(1)

Heap	*Hash Table*
Stack	*Binary Tree*

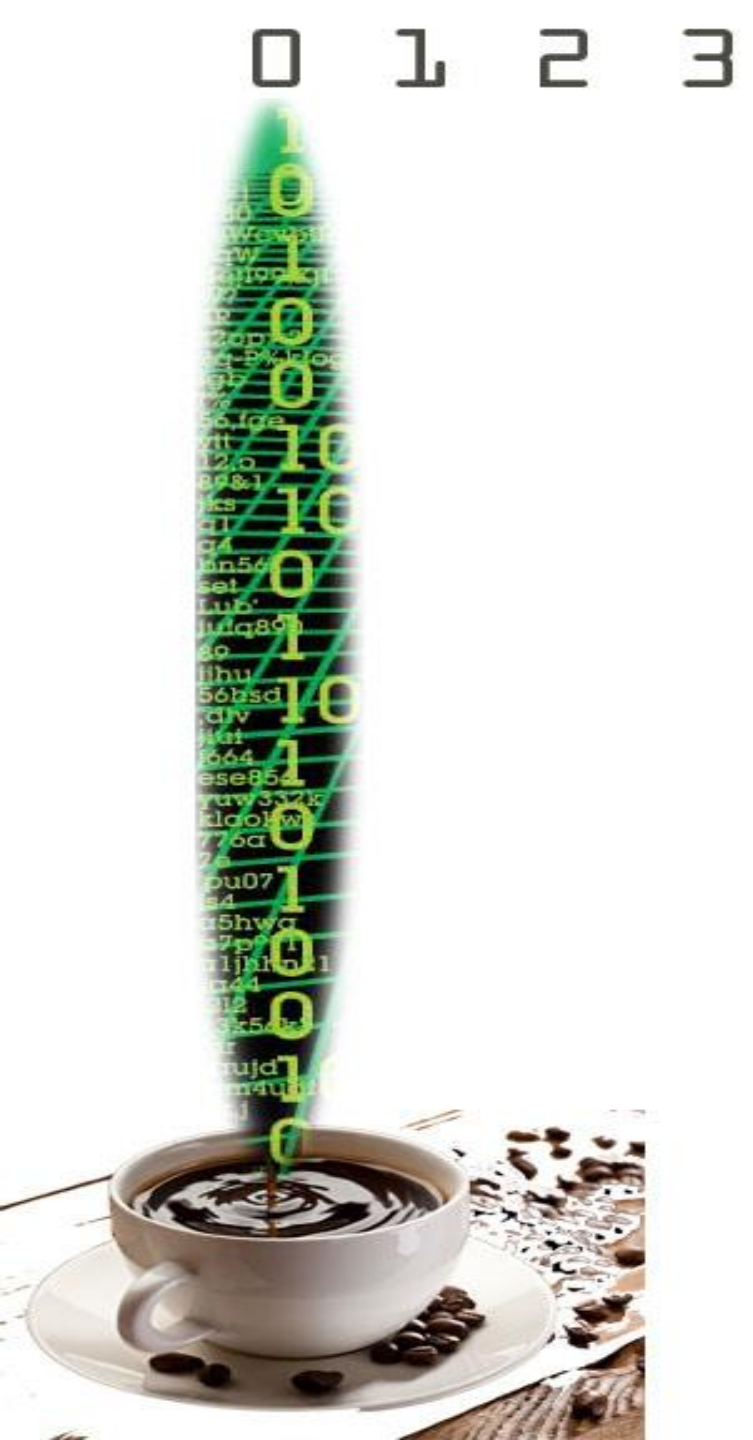
2000$ Question

```
int test(int x, int y)
{
   cout << x << y;
}
```

```
int main() {
   int val = 0;
   test(val++, val);
}
```

The printed values will be...

- *00*
- *01*
- *10*
- *Undefined*



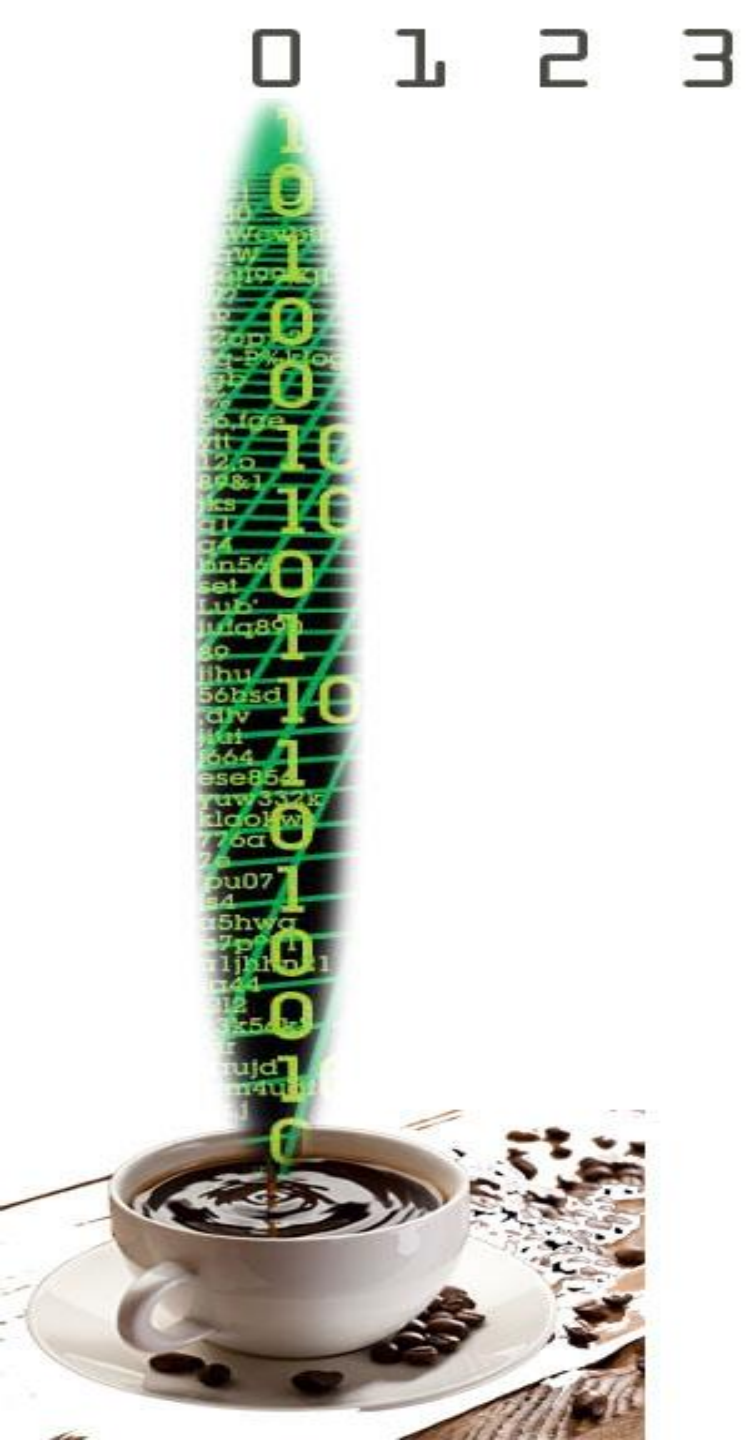
2000$ Question

```
int test(int x, int y)
{
   cout << x << y;
}
```

```
int main() {
   int val = 0;
   test(val++, val);
}
```

The printed values will be...

- *00*
- *01*
- *10*
- *Undefined*



2000$ Question

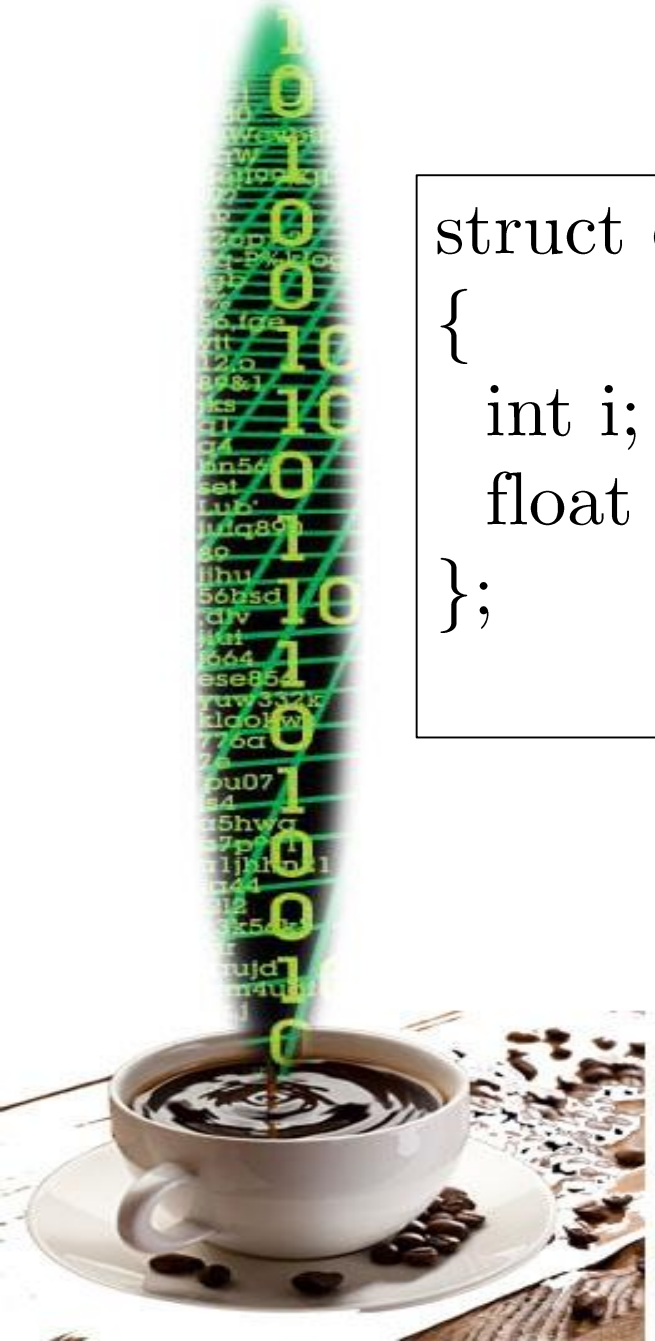
Passing parameters to a function is one thing.
The order of passing is another!

```
int main() {
    int val = 0;
    test(val++, val);
}
```

Stack Cleanup
Parameter Passing
Calling Convention

00	*01*
10	*It sucks... I know!*

0 1 2 3 4 5 6 7 8 9 A B C D E F



5000$ Question

```
struct data_struct
{
  int i;
  float f;
};
```

```
union data_union
{
  short s;
  float f;
  char c[3];
};
```

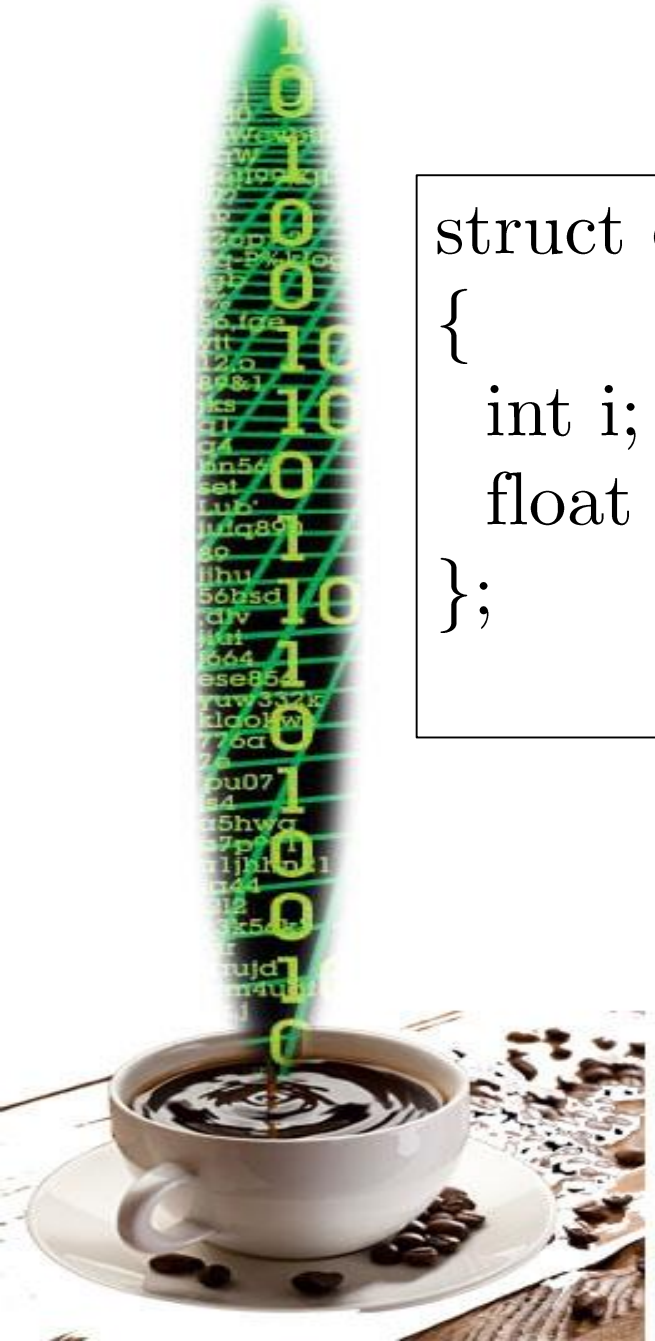
```
char data_char[5];

double data_dbl;
```

The variable that occupies the least data is...

- *data_struct*
- *data_union*
- *data_char*
- *data_dbl*

0 1 2 3 4 5 6 7 8 9 A B C D E F



5000$ Question

```
struct data_struct
{
  int i;
  float f;
};
```

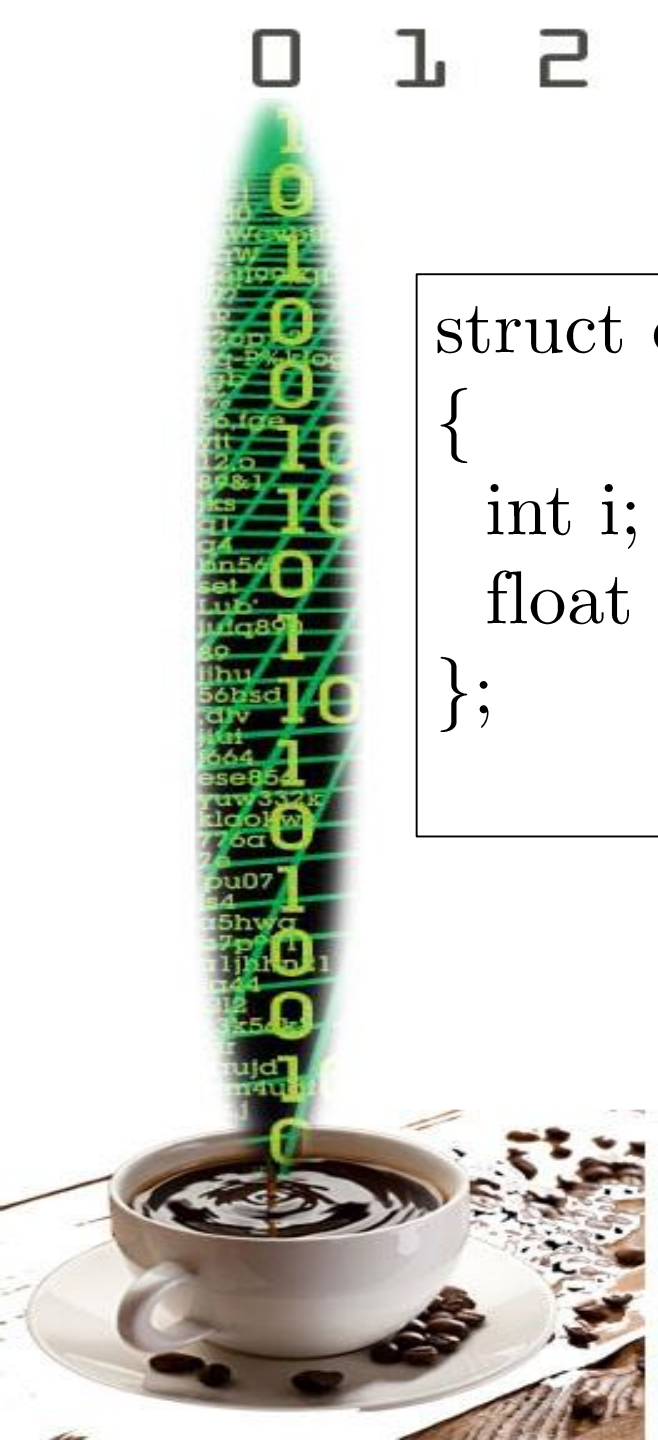
```
union data_union
{
  short s;
  float f;
  char c[3];
};
```

```
char data_char[5];

double data_dbl;
```

The variable that occupies the least data is...

- *data_struct*
- *data_union*
- *data_char*
- *data_dbl*



5000$ Question

```
struct data_struct
{
  int i;
  float f;
};
```

8 bytes

```
union data_union
{
  short s;
  float f;
  char c[3];
};
```

4 bytes

```
char data_char[5];
```

5 bytes

```
double data_dbl;
```

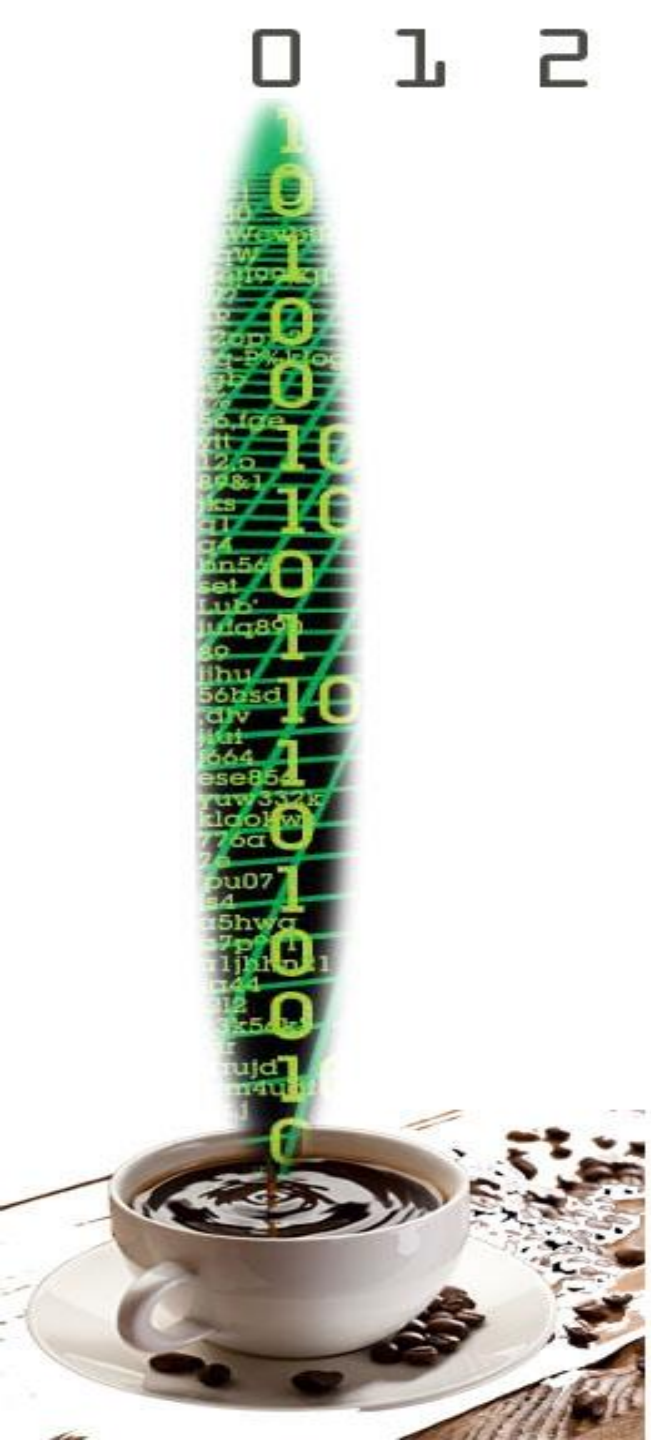
8 bytes

The variable that occupies the least data is...

- *data_struct*
- *data_union*
- *data_char*
- *data_dbl*

Bonus Question

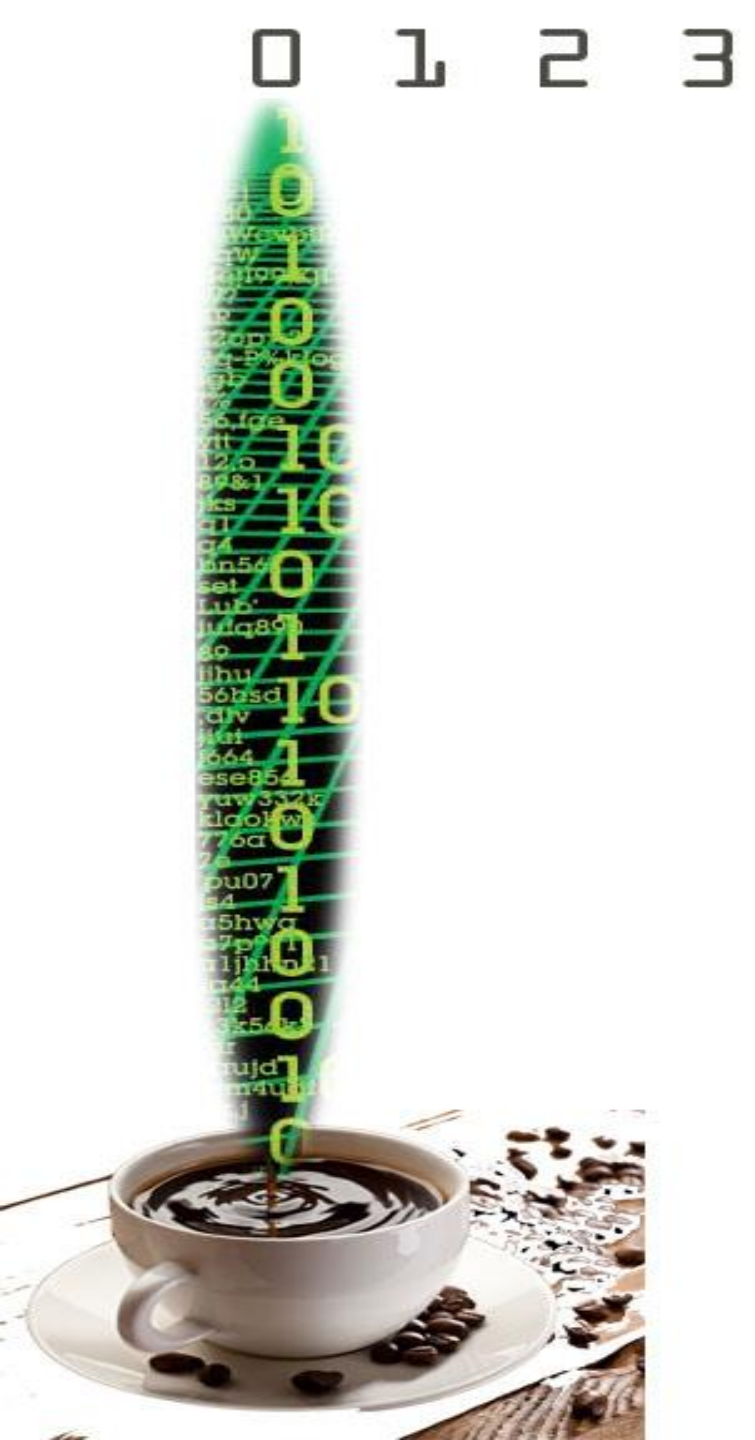
Write a one-liner to swap
two variables.



Bonus Question

Write a one-liner to swap two variables.

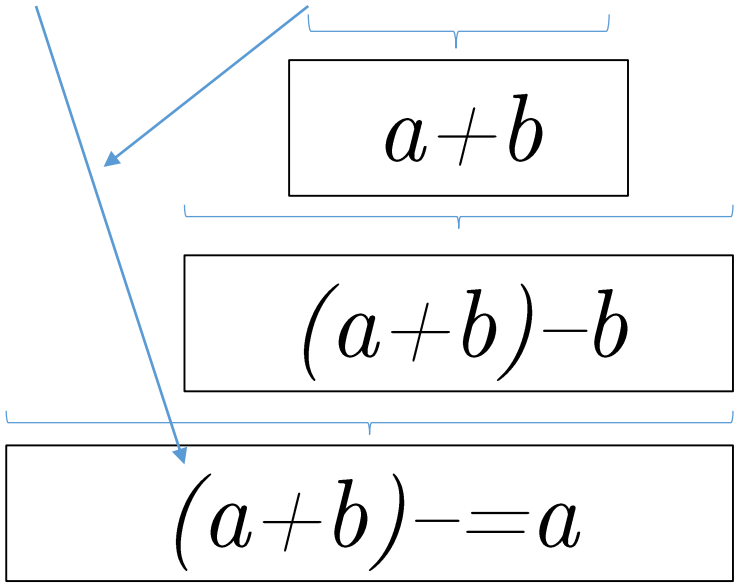
$$a-=b=(a+=b)-b;$$



Bonus Question

Write a one-liner to swap two variables.

$$a-=b=(a+=b)-b;$$



a	b
$a+b$	b
$a+b$	a
b	a

Win32 API – Basic Structures

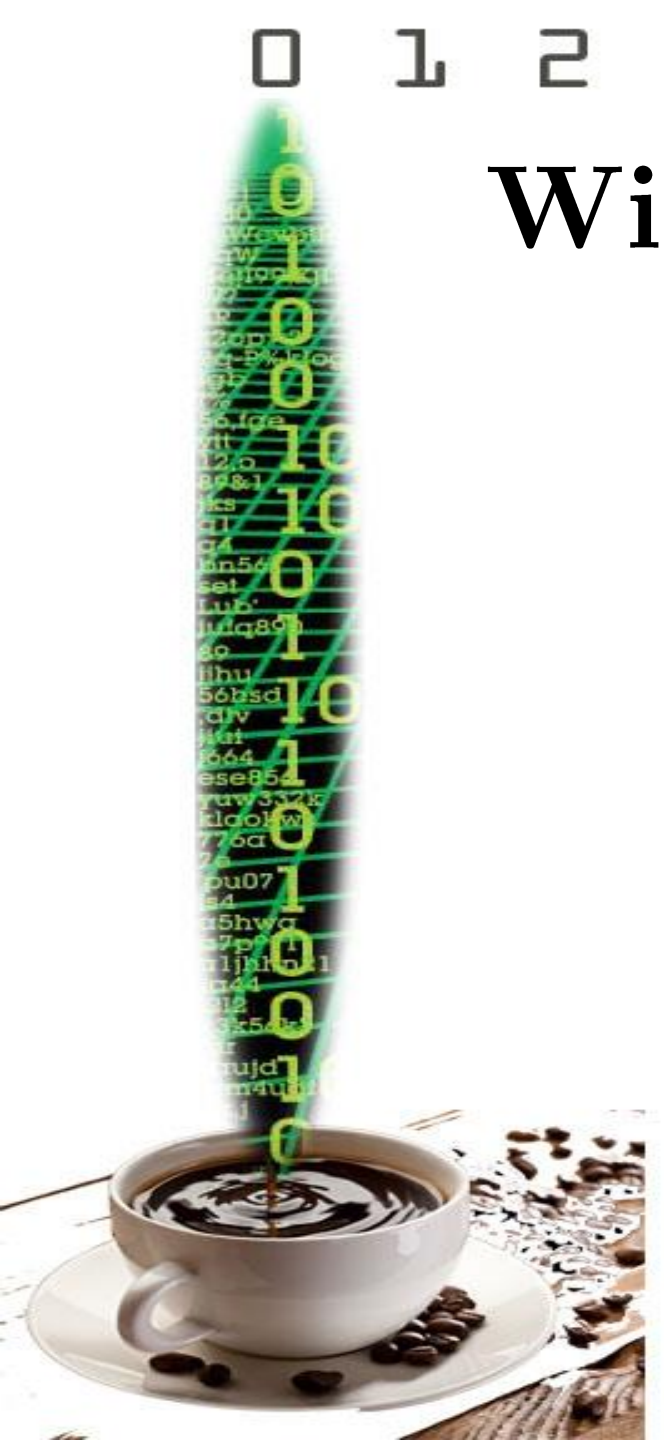
Hundreds, actually...

- Window → struct WNDCLASSEX
- Message → struct MSG
- Controls
- Other (e.g. OPENFILENAME in commdlg.h)

Win32 API – Basic Structures (2)

Example

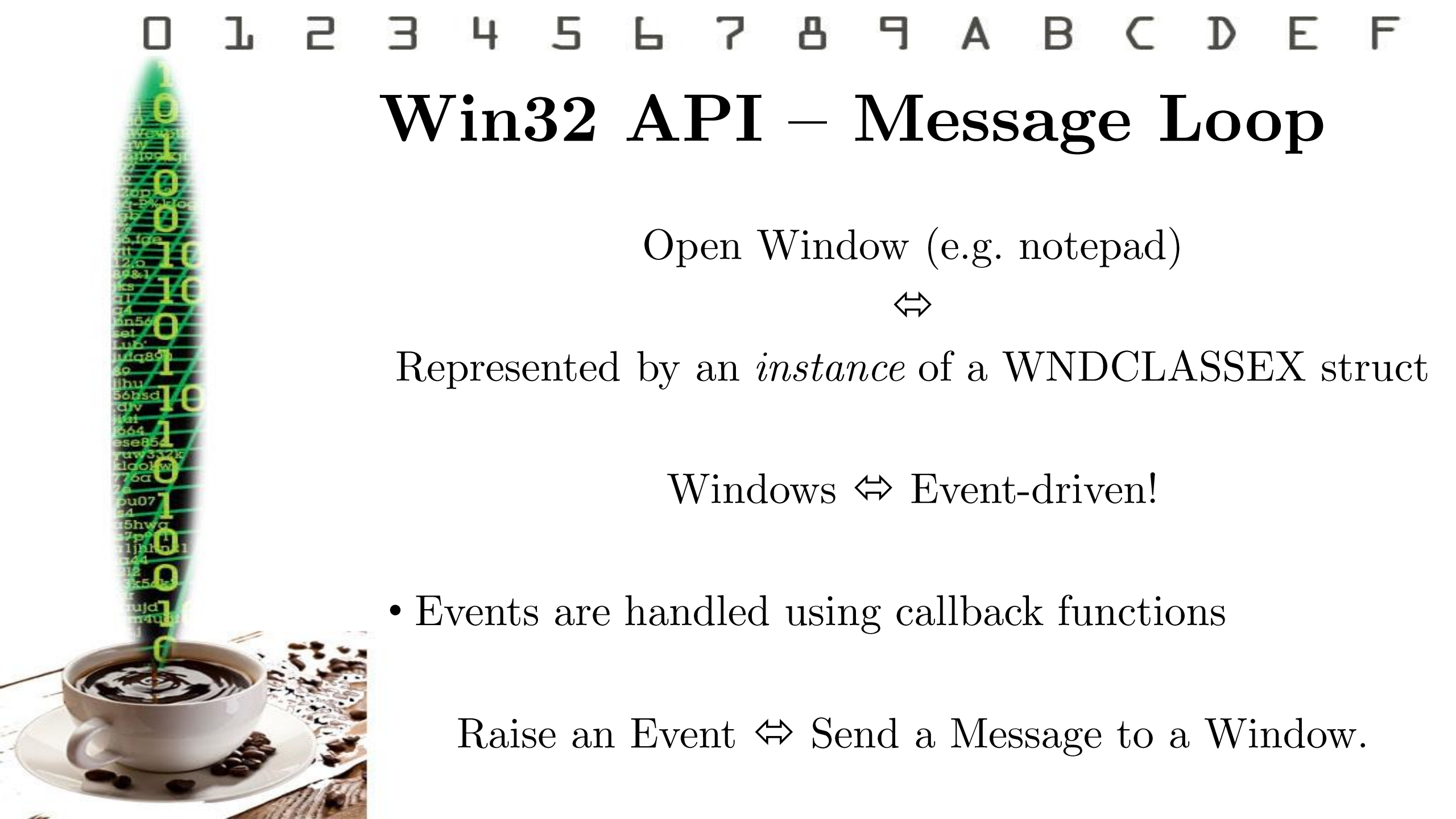
```
typedef struct tagMSG {
        HWND hwnd;
        UINT message;
        WPARAM wParam;
        LPARAM lParam;
        DWORD time;
        POINT pt;
} MSG,*LPMSG,*PMSG;
```



Win32 API – Basic Structures (3)

Structures are used as "classes"

- They store information about objects flying around!
  - Windows, files, cursors, etc.
- Windows *knows* what to do with them.
- Win32 API is a collection of **structures** and **functions** to "fool around" with these structures.



Win32 API – Message Loop

Open Window (e.g. notepad)

$\Leftrightarrow$

Represented by an *instance* of a WNDCLASSEX struct

Windows $\Leftrightarrow$ Event-driven!

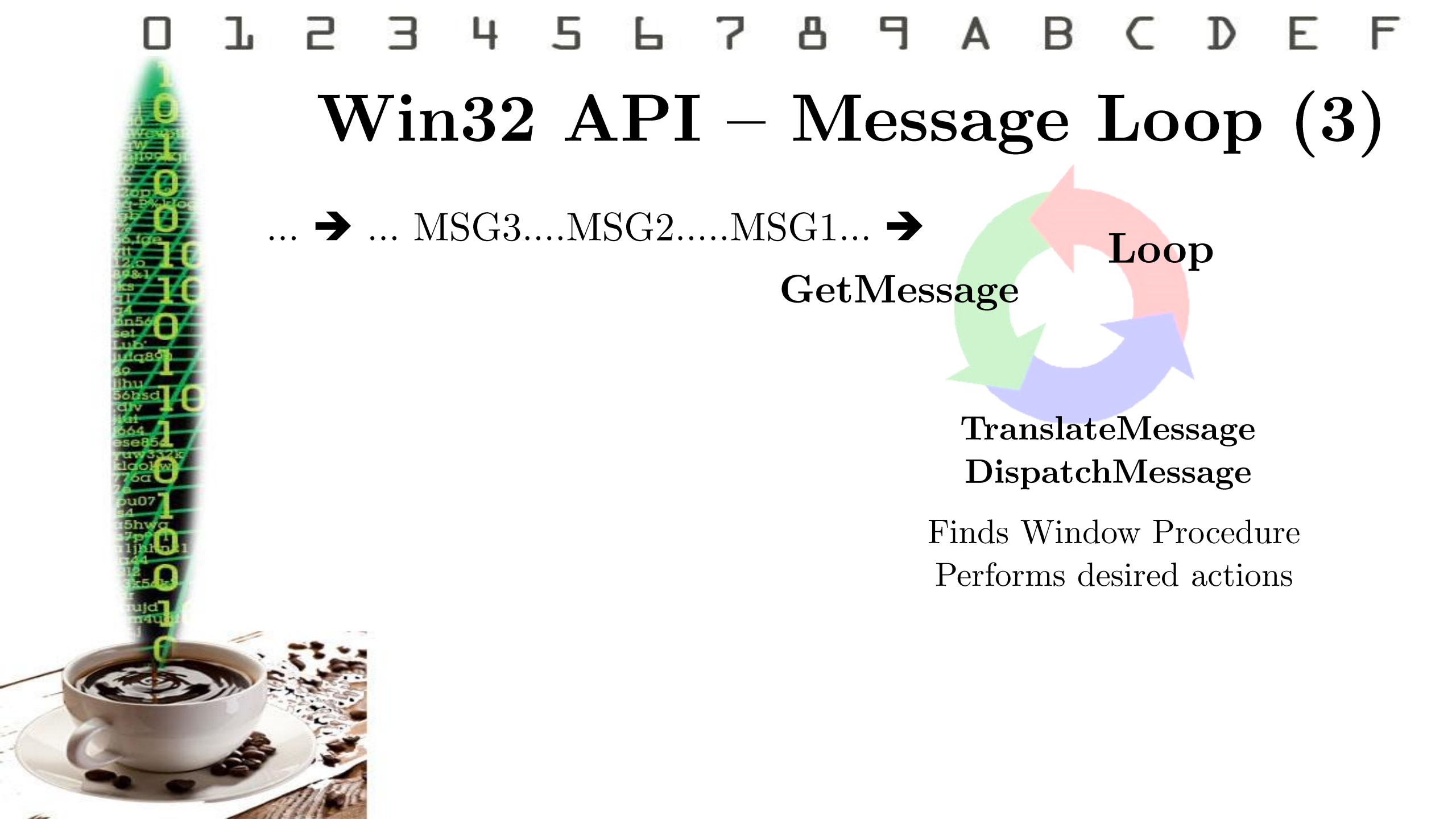
- Events are handled using callback functions

Raise an Event $\Leftrightarrow$ Send a Message to a Window.

Win32 API – Message Loop (2)

... ➔ ... MSG3....MSG2.....MSG1... ➔

A window's **Message Loop**



Win32 API – Message Loop (3)

... ➔ ... MSG3....MSG2.....MSG1... ➔

Loop

GetMessage

TranslateMessage

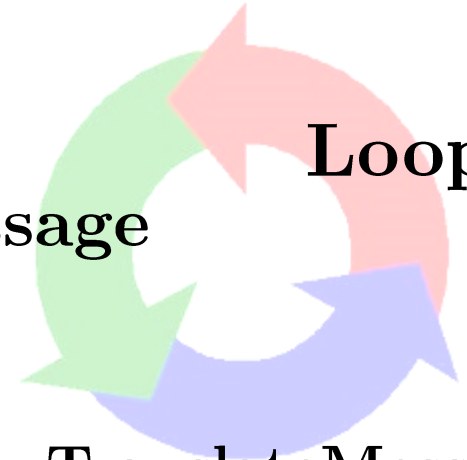
DispatchMessage

Finds Window Procedure

Performs desired actions

Win32 API – Message Loop (4)

... ➔ ... MSG3....MSG2.....MSG1... ➔



Loop

GetMessage

Message from Message Queue

➔

Message Loop

➔

Suitable Handle Function

➔

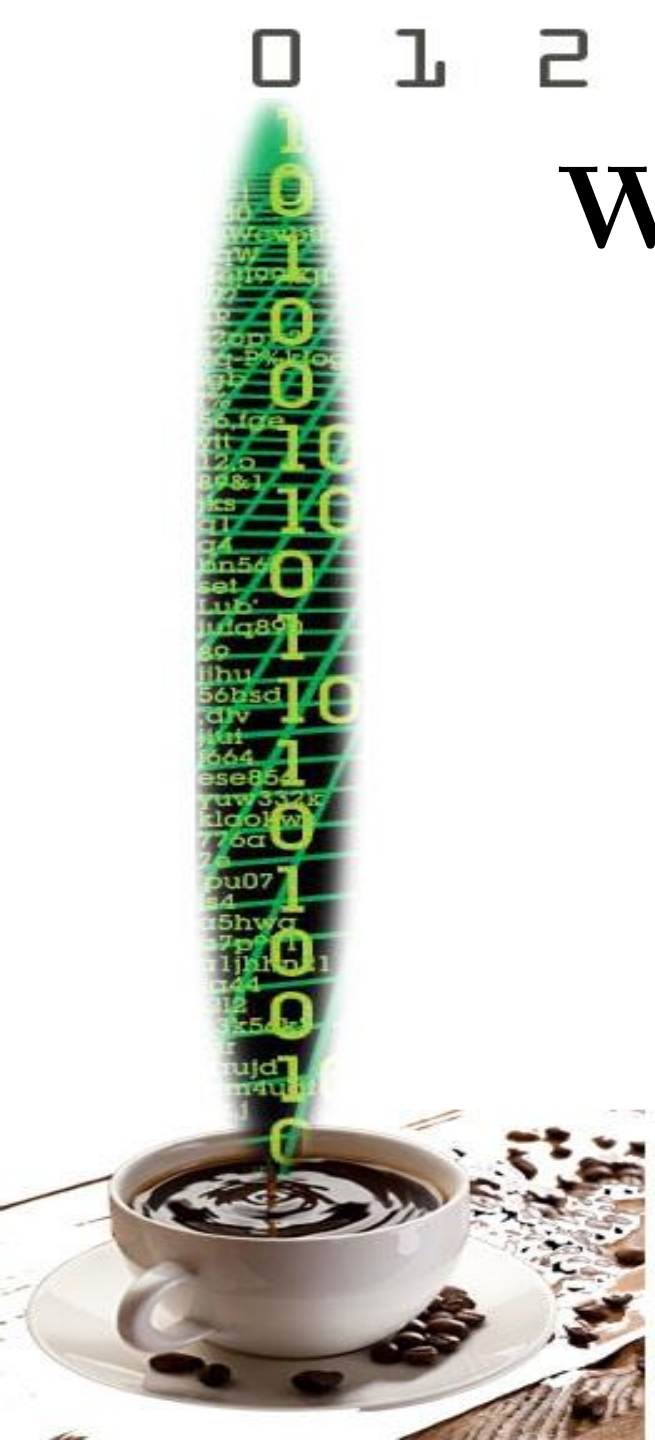
Next Message from Queue

TranslateMessage

DispatchMessage

Finds Window Procedure

Performs desired actions



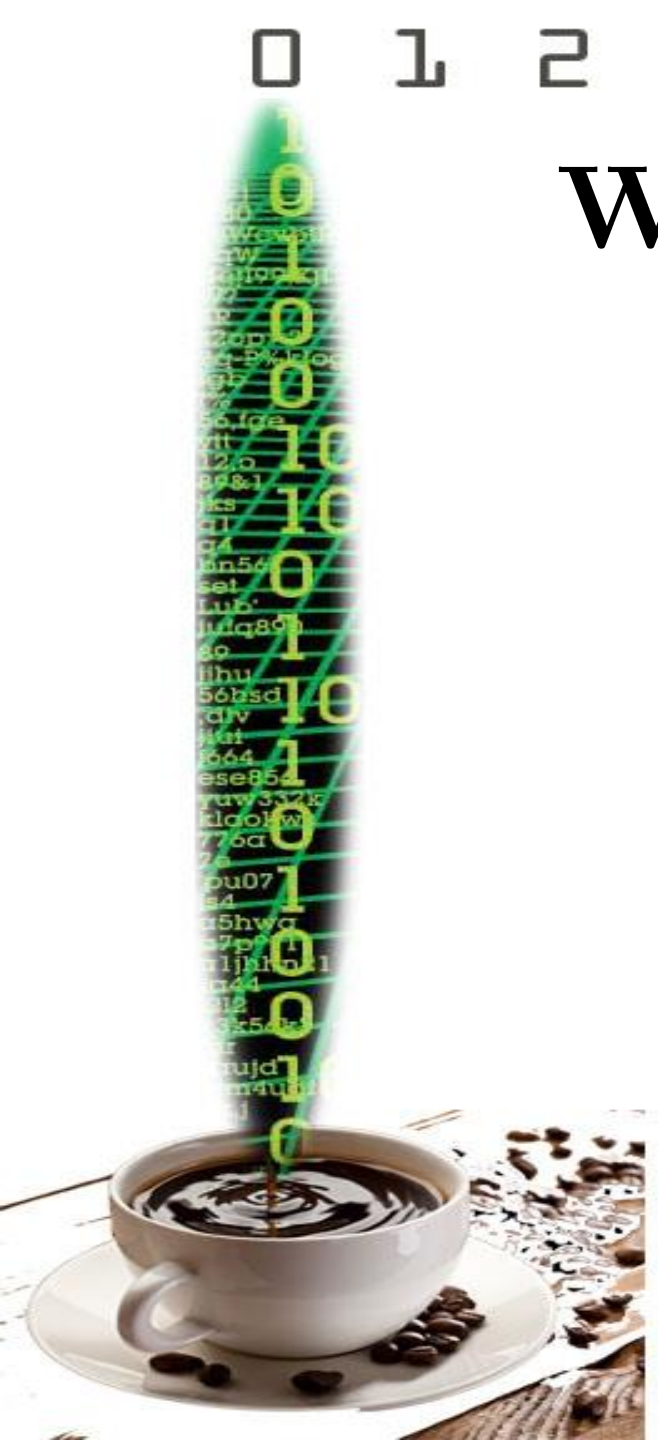
Win32 API – Message Loop (5)

Oh, and Messages in Win32 API are merely...

Numbers

e.g.

```
#define WM_DESTROY          2
#define WM_COMMAND          273
#define IDC_OPEN_BUTTON     1001
```

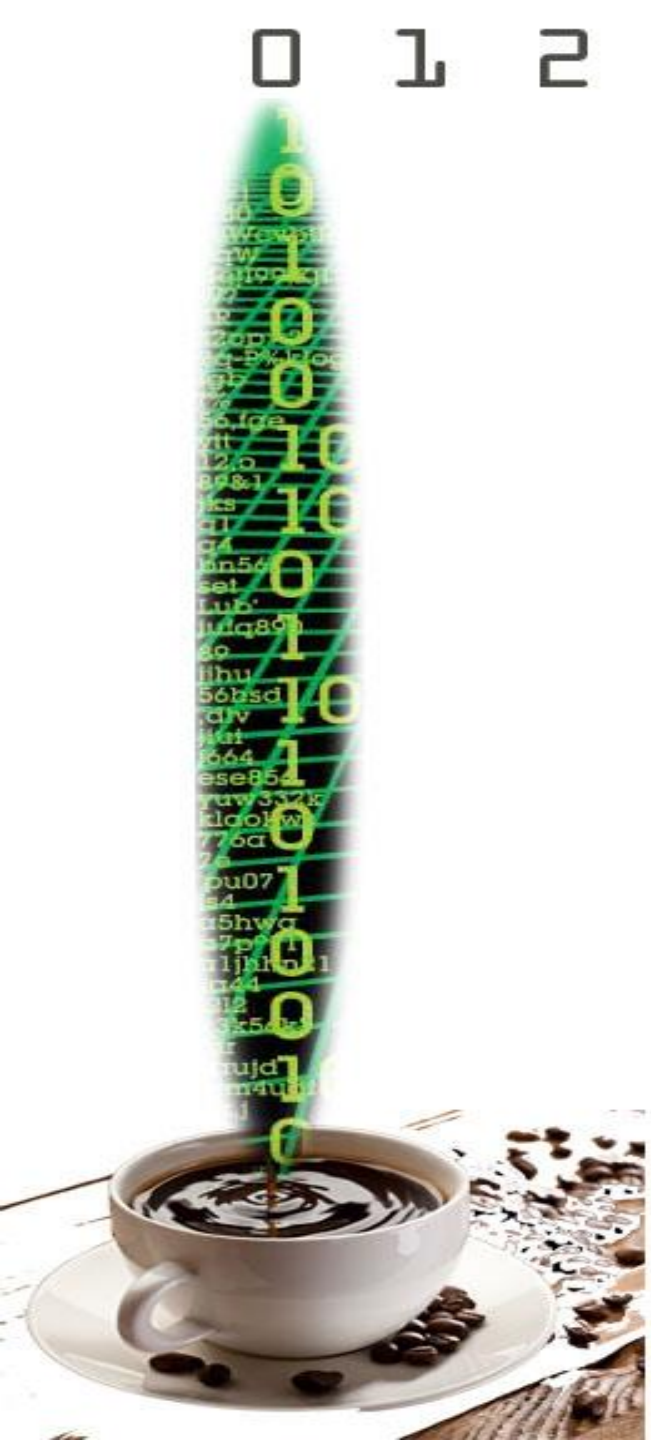


Win32 API – Message Loop (6)

When a Window is created, a Message Loop is created too.

The declaration looks like this:

```
LRESULT CALLBACK WindowProcedure
    (HWND, UINT, WPARAM, LPARAM) {
    // Stuff to perform loop management.
}
```



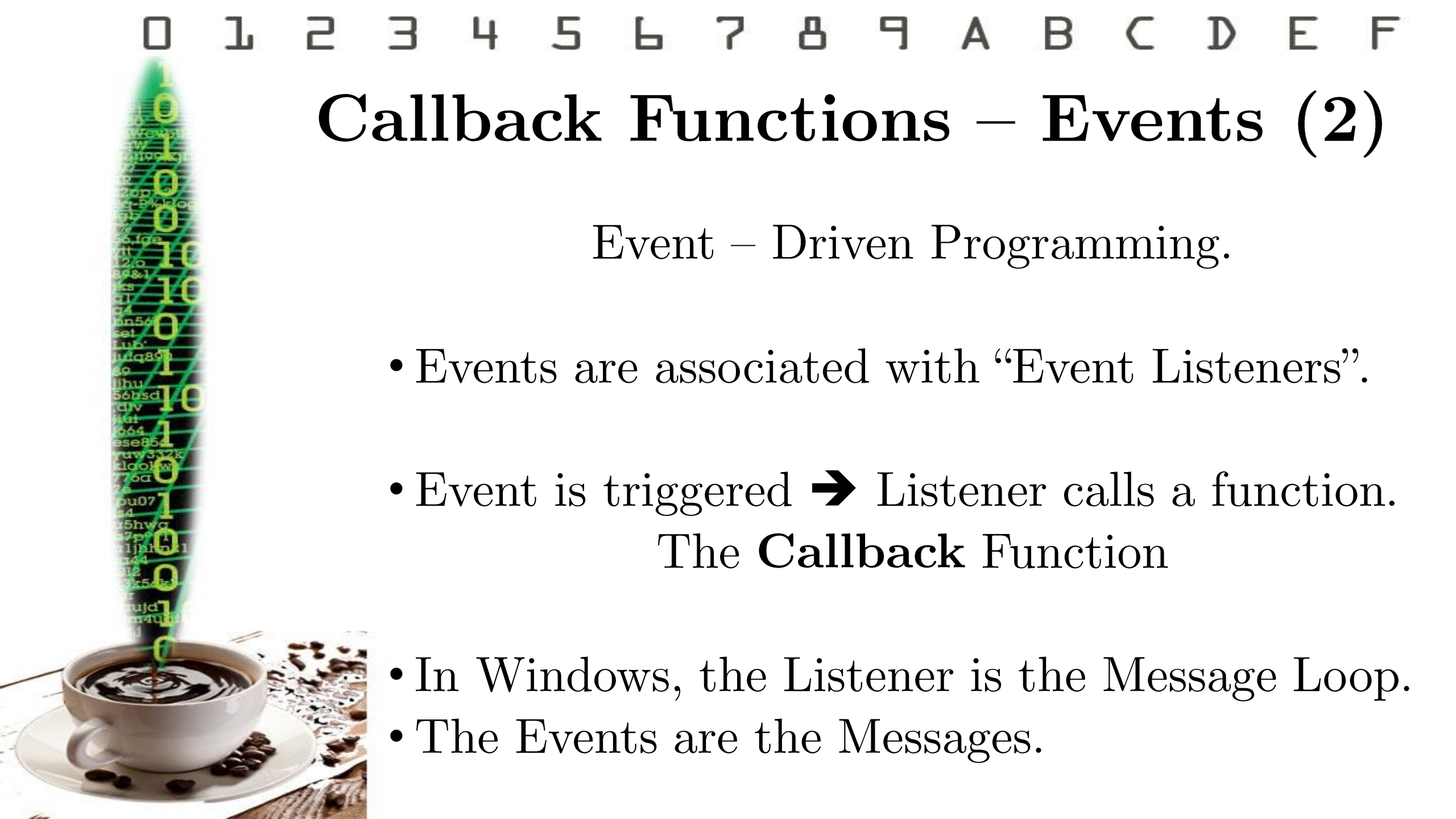
Callback Functions - Events

Event – Driven Programming.

- *Controls* provide functionality through *Events*.

- Events are (usually) User Interactions.

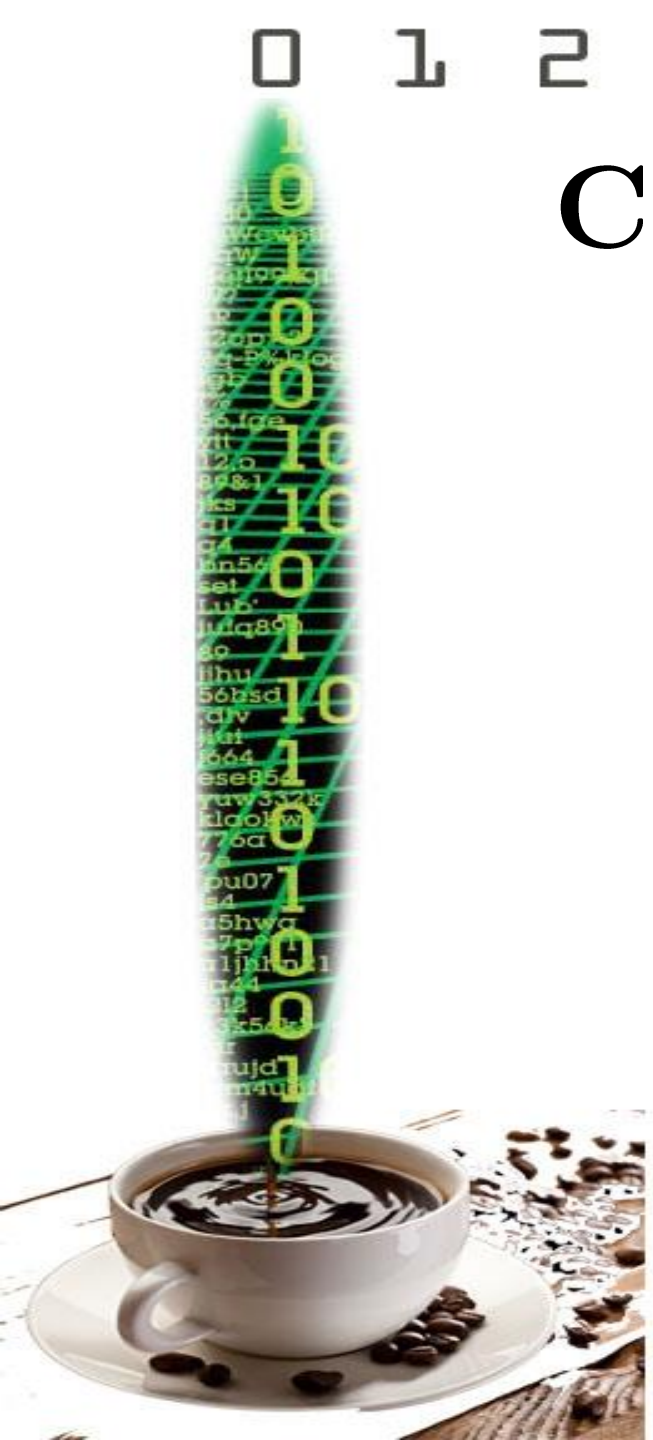
  Click, Keypress, Mouse Move, etc...



Callback Functions – Events (2)

Event – Driven Programming.

- Events are associated with “Event Listeners”.
- Event is triggered ➔ Listener calls a function.
  The **Callback** Function
- In Windows, the Listener is the Message Loop.
- The Events are the Messages.

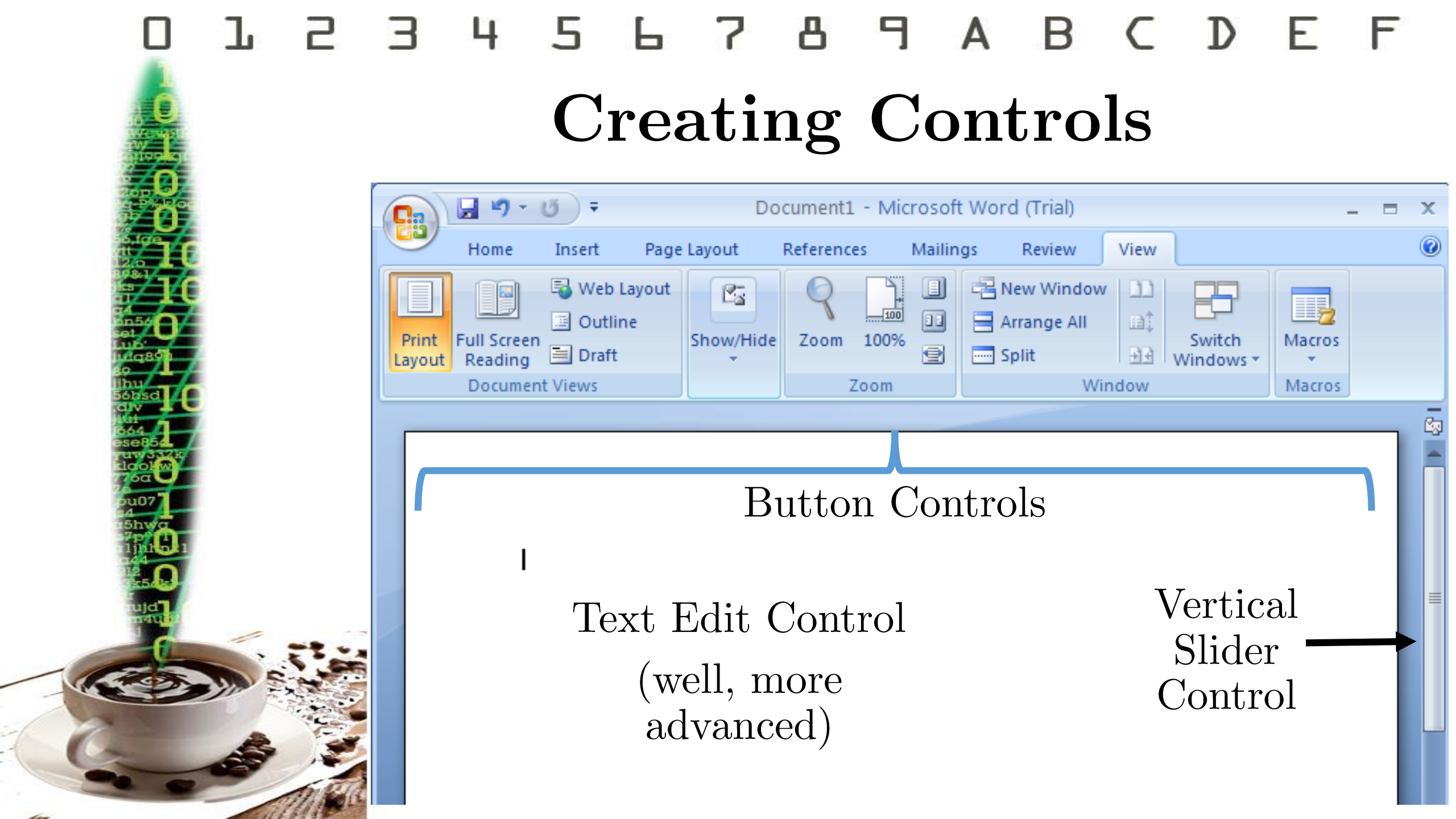


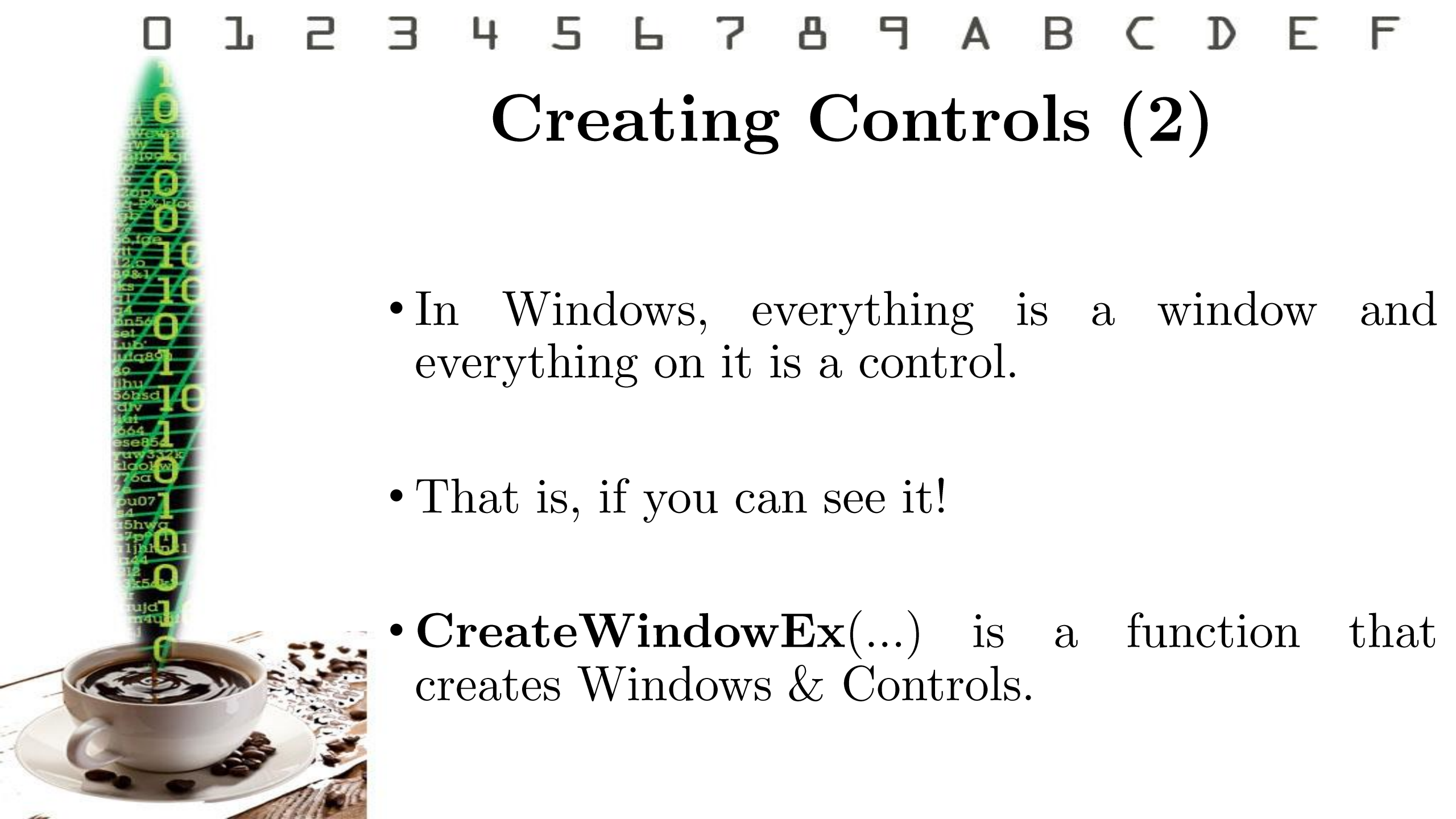
Callback Functions – Events (3)

Simple Contrived Example

- Create a button named btnRun.
- Create a function DoSomething().
- Assign DoSomething() to btnRun.onClick.

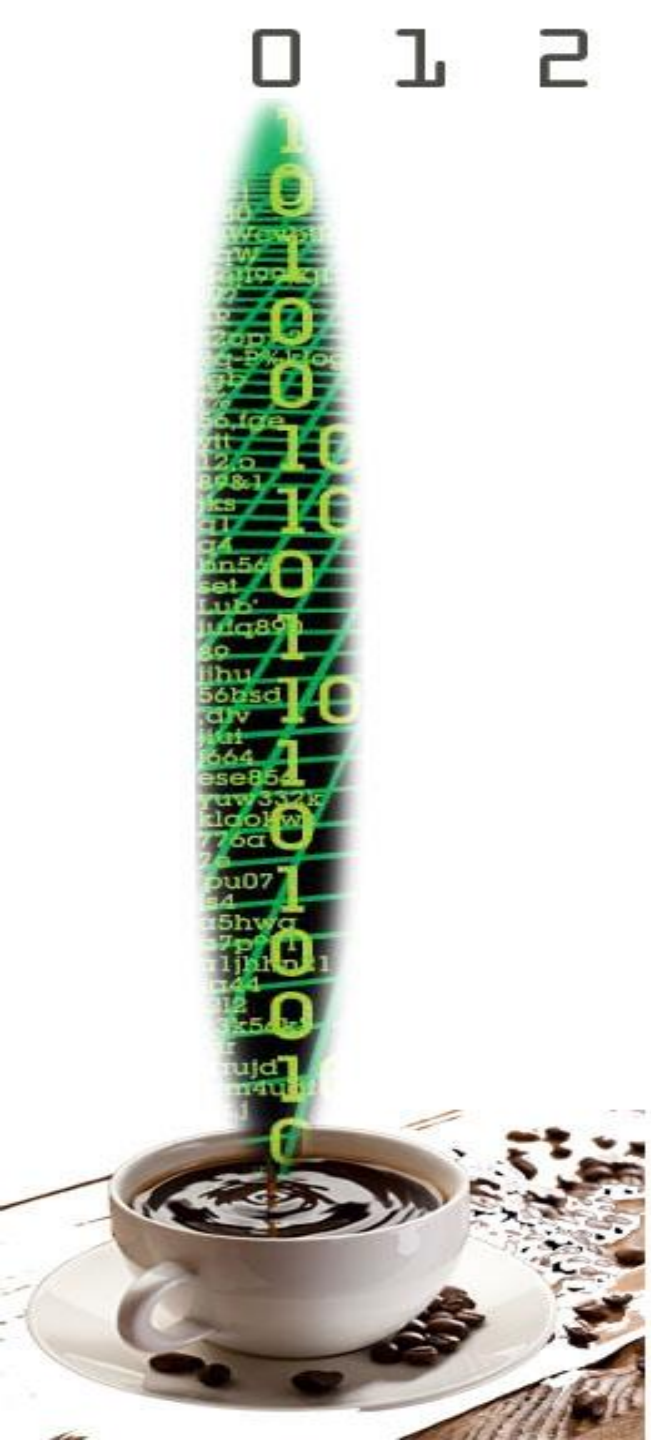
- Create another function changeColor().
- Assign changeColor() to btnRun.onMouseOver.


0 1 2 3 4 5 6 7 8 9 A B C D E F
Creating Controls
Document1 - Microsoft Word (Trial)
Home
Insert
Page Layout
References
Mailings
Review
View
Print Layout
Full Screen Reading
Web Layout
Outline
Draft
Document Views
Show/Hide
Zoom
100%
Zoom
New Window
Arrange All
Split
Switch Windows
Window
Macros
Macros
Button Controls
Text Edit Control
(well, more advanced)
Vertical Slider Control



Creating Controls (2)

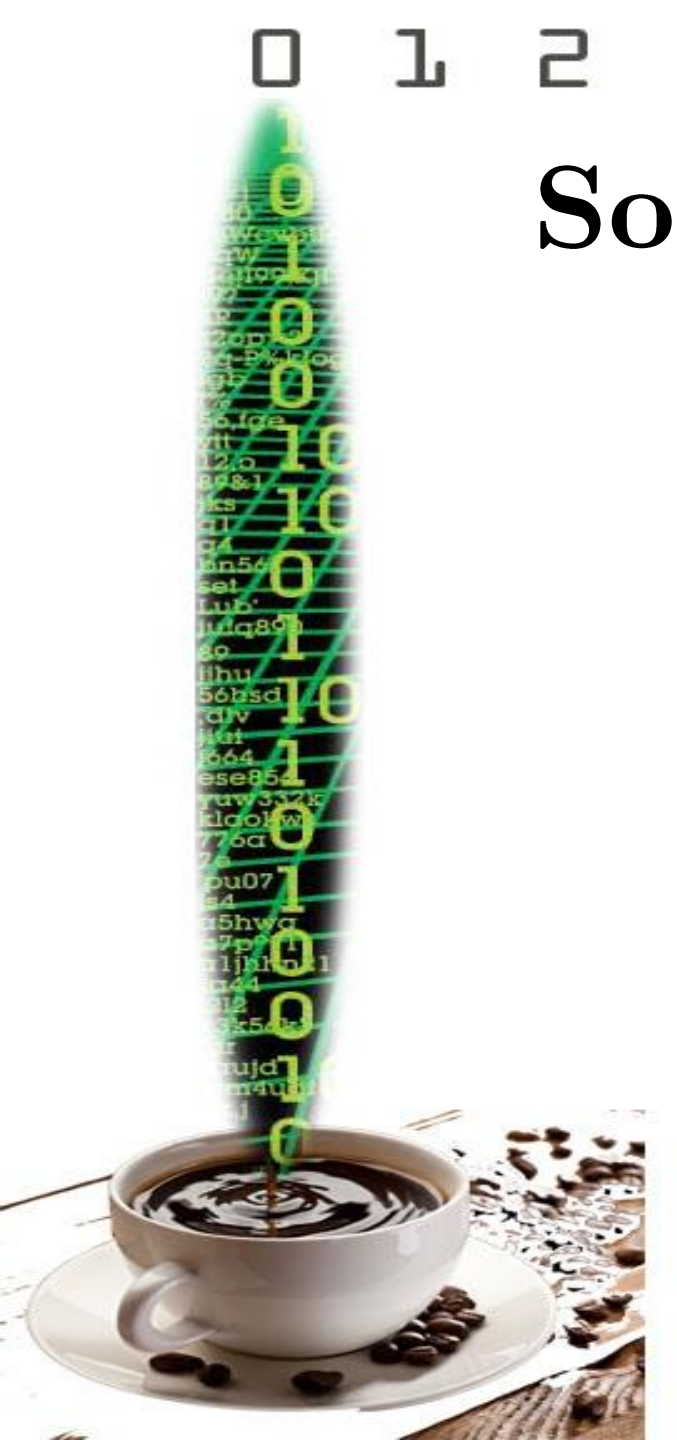
- In Windows, everything is a window and everything on it is a control.
- That is, if you can see it!
- **CreateWindowEx**(...) is a function that creates Windows & Controls.



Creating Controls (3)

- When a window is created, it is assigned:
  (***among others***)

- Classname.
- Title (if it is a window rather than a control).
- Style (aspects of appearance).
- Position on Screen.
- Width and Height.
- Parent Window.
- Menu or IDC (Control Identifier).

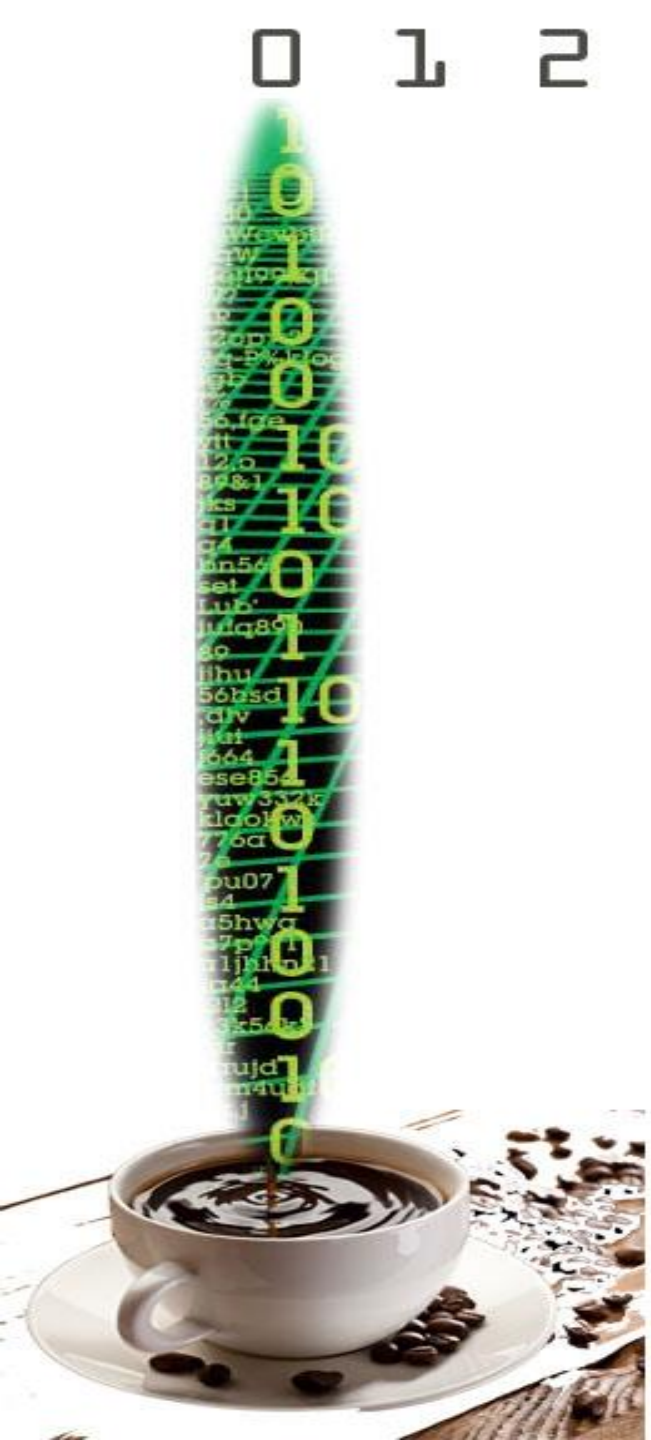


So... a Windows Application is...

- A window drawn on the screen.
- Controls drawn on the window.

  Buttons, textboxes, menu title, ribbons, control containers, etc.
- Functionality (usually in libraries) assigned to the controls.

  Through Message Loop callback function handlers.

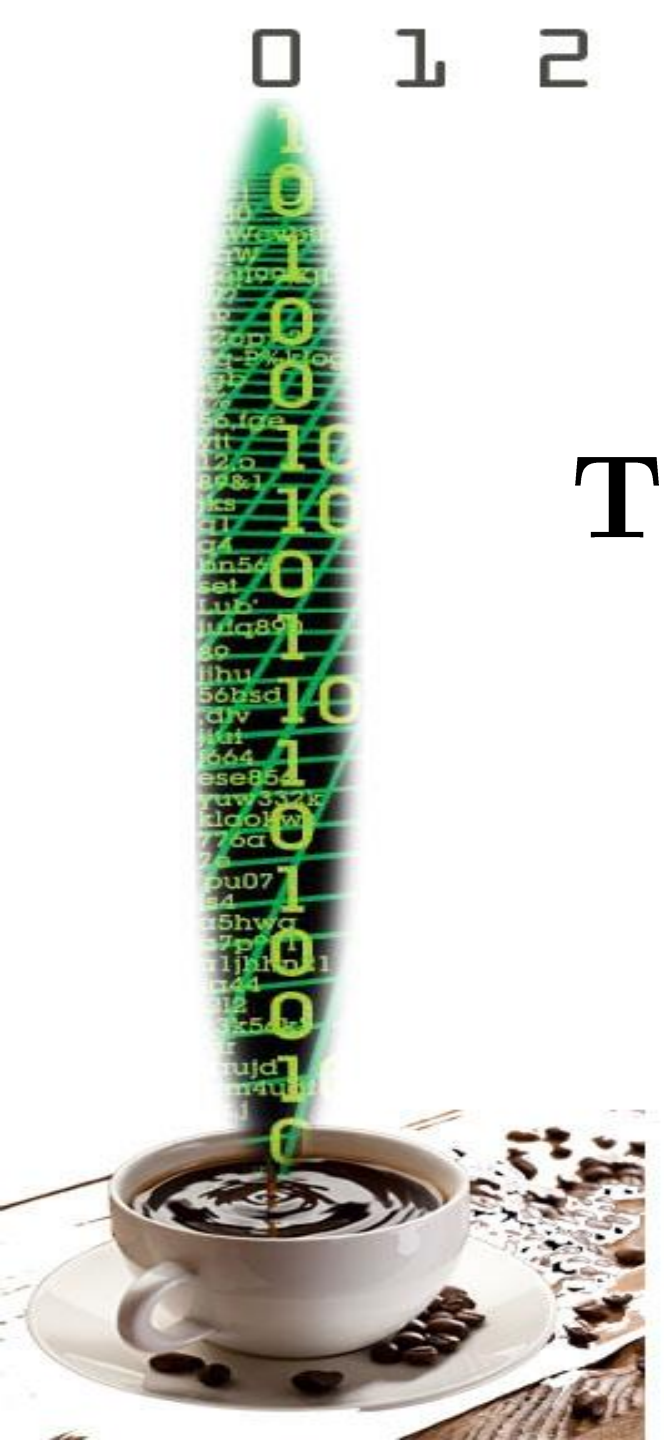


In The Next Episode

- Formatted Text Input/Output with C++

- Regular Expressions in Programming

  (regular languages, anyone? rational, maybe?)



Thank you for your attention!

Time to code...