Algoritmi e Strutture Dati

Lezione 1

www.iet.unipi.it/a.virdis

Antonio Virdis

antonio.virdis@unipi.it

“Ma io so già programmare!”

Fondamenti I

Sia dato un array contenente delle **frasi**.

Scrivere un programma che prenda in input una **stringa** e restituisca tutte le frasi che la contengono.

Fondamenti II

Realizzare un motore di ricerca che abbia complessità O(log n) sulle operazioni di lettura.

Algoritmi e **Strutture Dati**

Informazioni

- Lezioni teorico-pratiche
- Esercizi assegnati per casa
- Esercitazioni tramite portale online
- Esame in laboratorio

Pre Requisiti

- Fondamenti I
- Utilizzo compilatore
- Comandi base unix

Sommario

- Ordinamento: Insertion Sort
- Standard Template Library
- Debug

esperimenti

- Implementare **InsertionSort**
- Chiedere a **InsertionSort** come funziona
- Analizzare casi limite
- Prestazioni

Ordinamento

Stampa

```
// stampa i "len" valori contenuti in arr,
// separati da tabulazioni
     stampaArray( int arr[] , int len );



/*Scrivere un programma che:
  - possa memorizzare 10 interi
  - memorizzi 10 interi
  - usi una funzione per stampare i 10 interi     */
int main()
{
     const int sSize = 10;
     int sArray[sSize] = { 9,5,1,14,0,9,5,1,14,0 };


}
```

Stampa

```
1   #include <iostream>
2   Using namespace std;
3
4   void stampaArray( int arr[] , int len )
5   {
6       for( int i=0 ; i < len ; ++i )
7           cout << arr[i] << "\t" ;
8       cout << endl;
9   }
10
11
12  int main( )
13  {
14      const int sSize = 10;
15      int sArray[sSize] = { 9,5,1,14,0,9,5,1,14,0 };
16      stampaArray(sArray,sSize);
17      return 1;
18  }
```

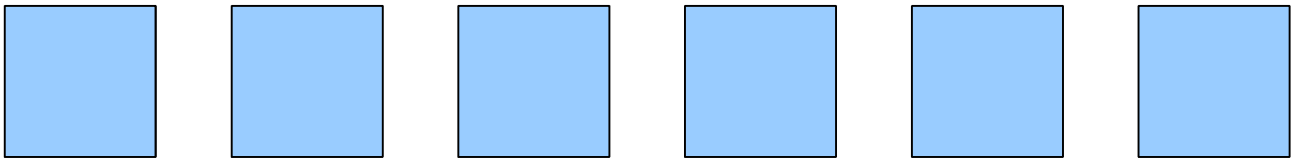
Nomi delle Variabili

prova pippo

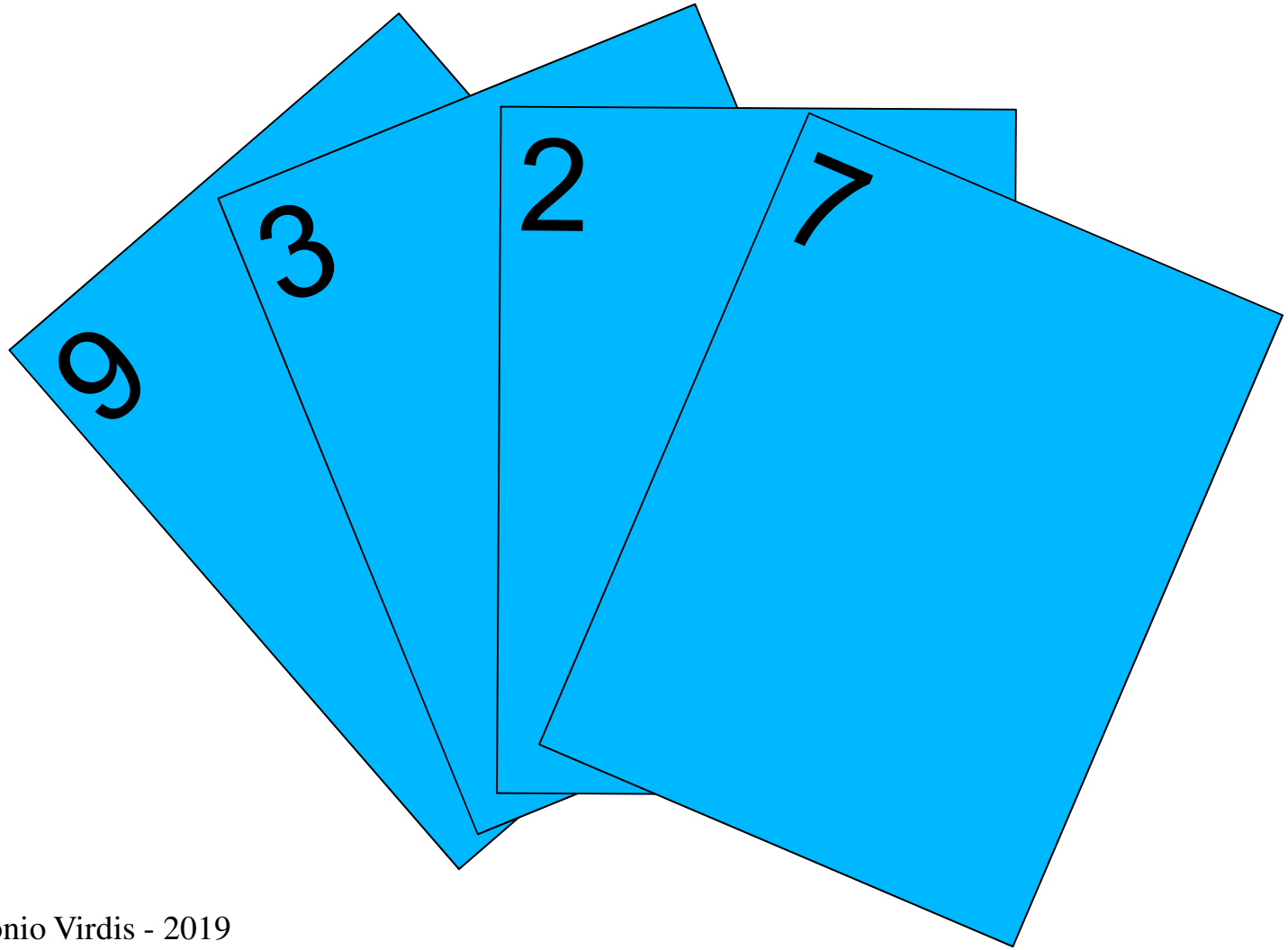
```
1
2
3
4
5
6
7   int main( ){const int a = 10;
8   int pippo[a] = { 9,5,1,14,0,9,5,1,14,0 };
9   prova(pippo,a);
10  return 1;}
11
12
13
14
15
16
17
18
```

?

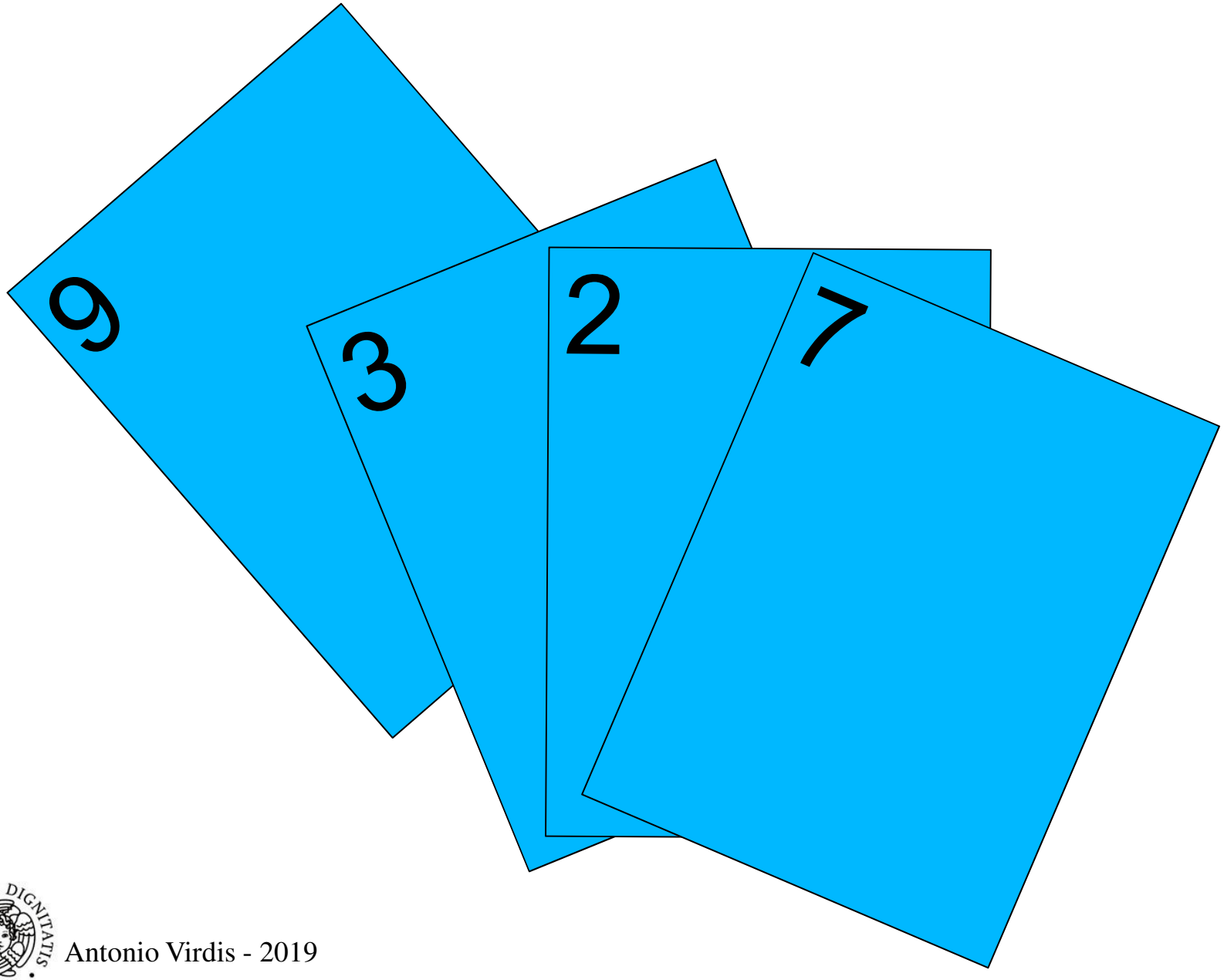
Insertion Sort: rappresentazione



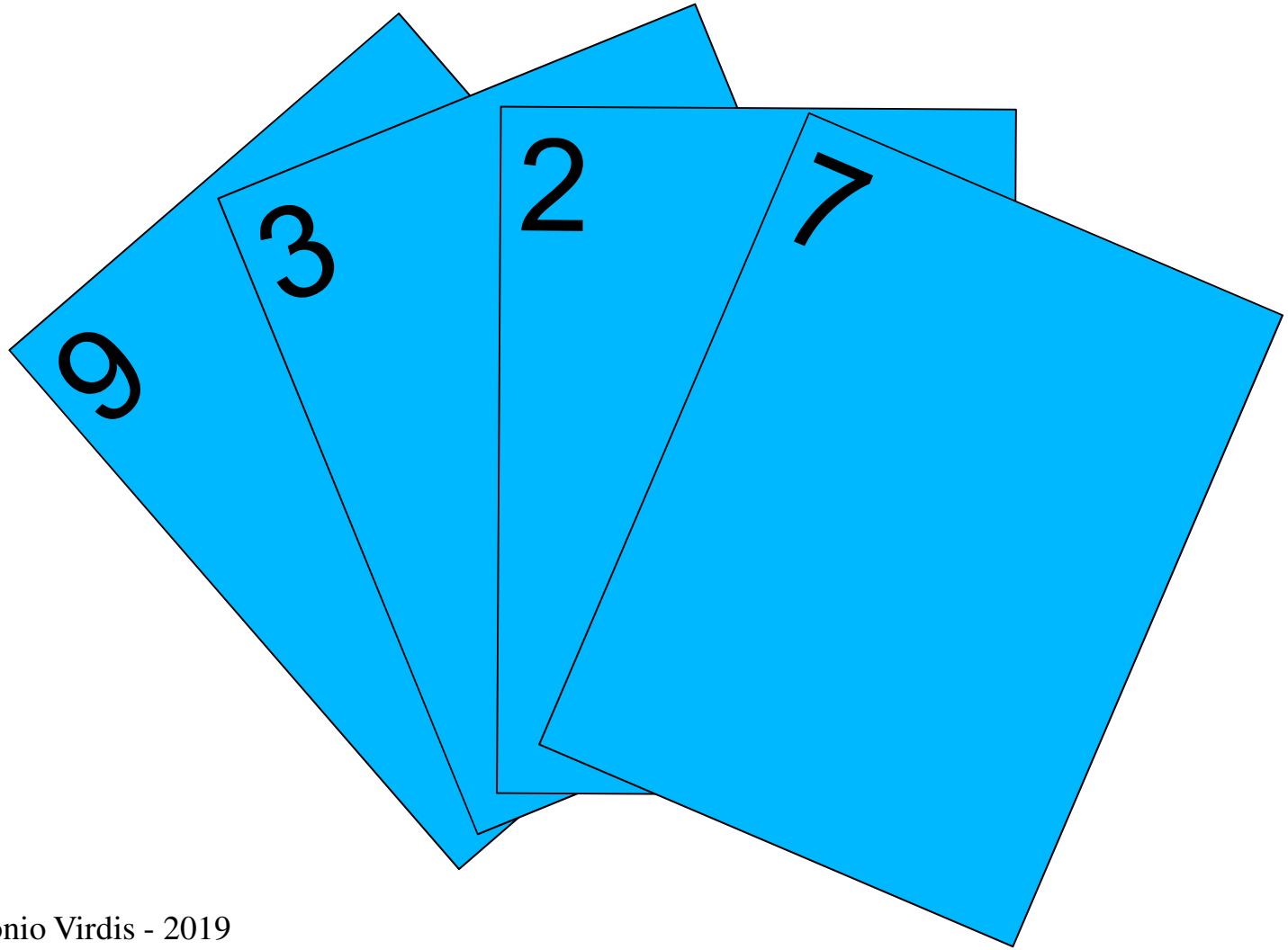
Ordinare Carte da Gioco



Ordinare Carte da Gioco

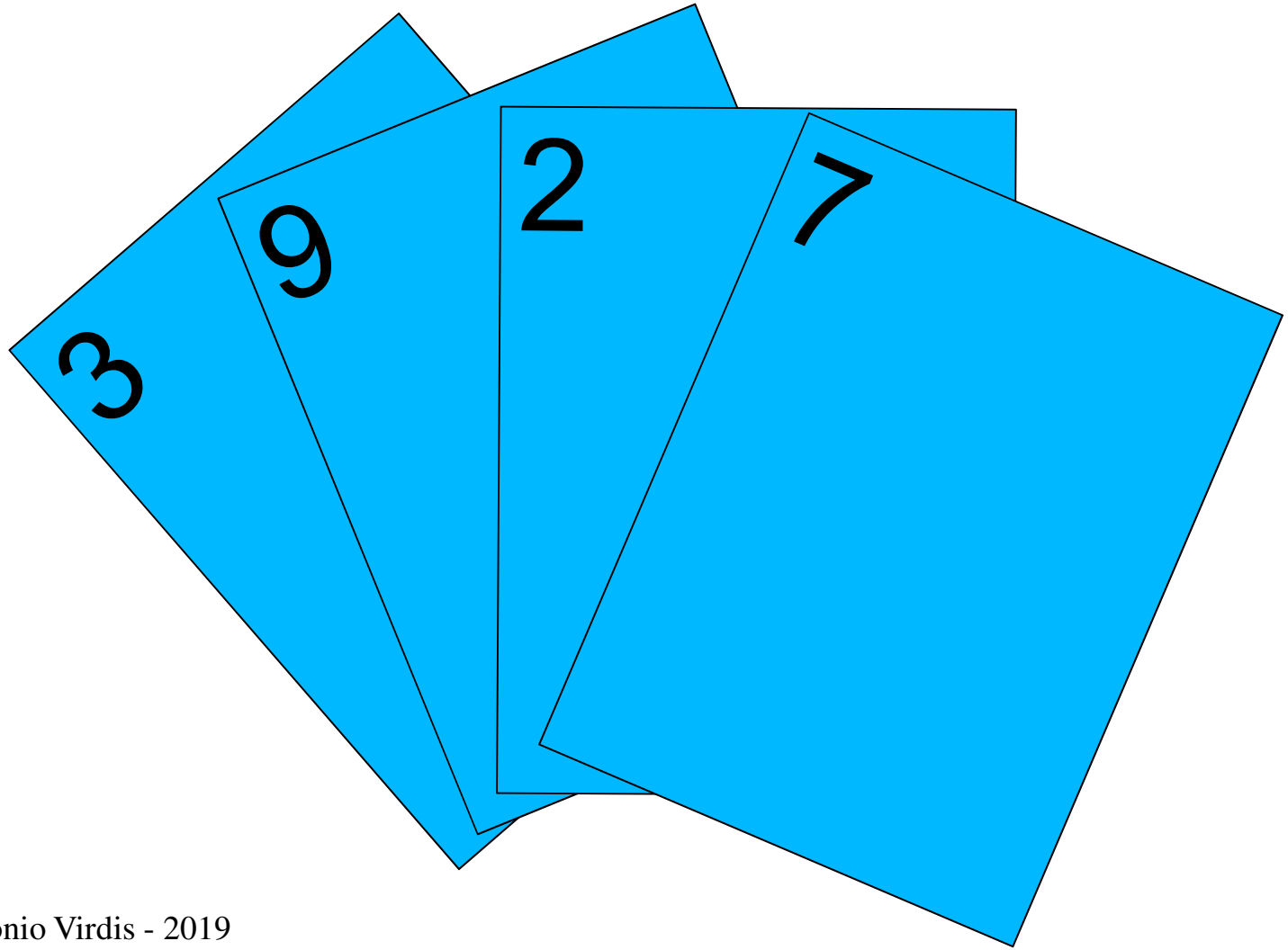


Ordinare Carte da Gioco

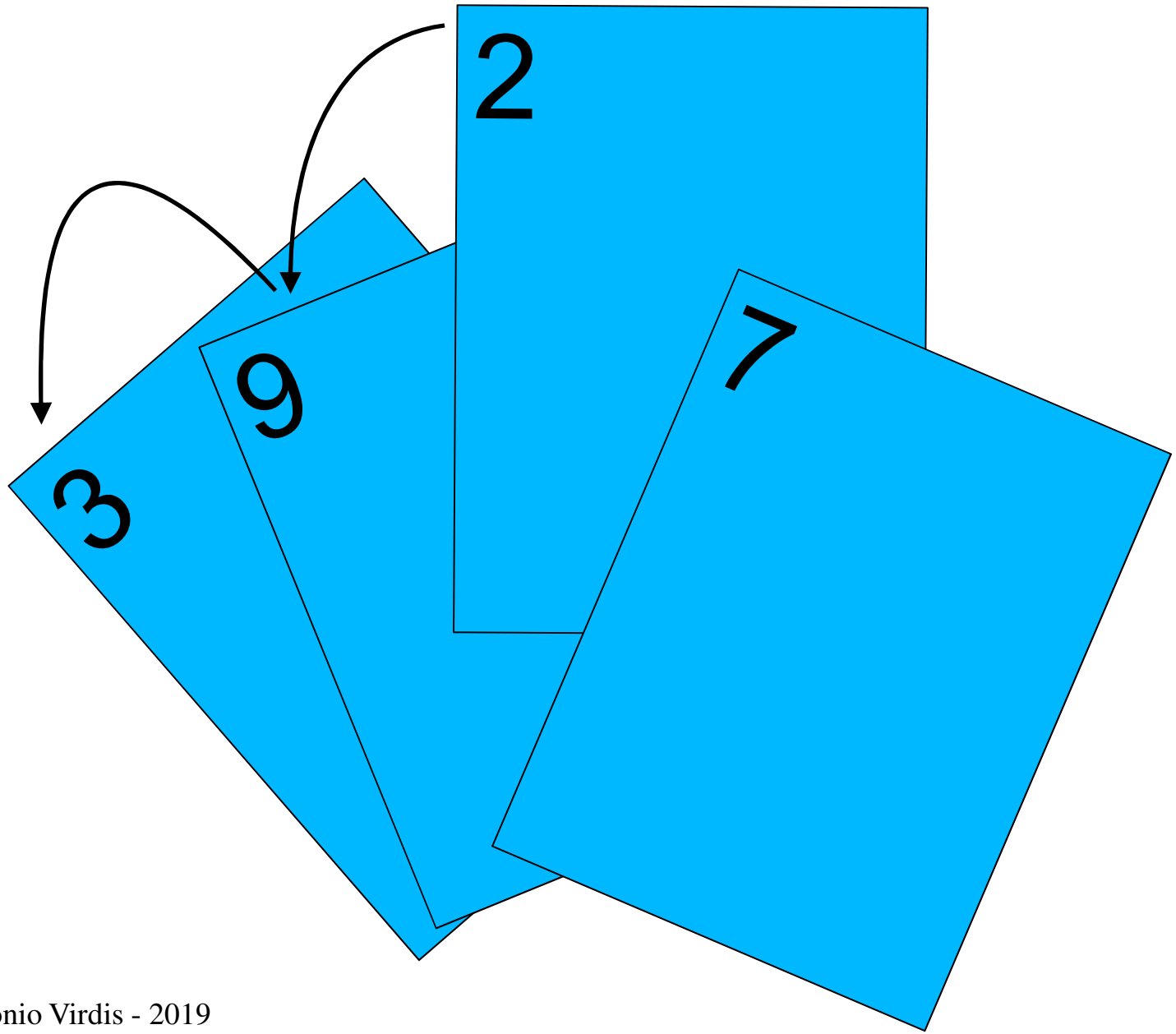


Ordinare Carte da Gioco

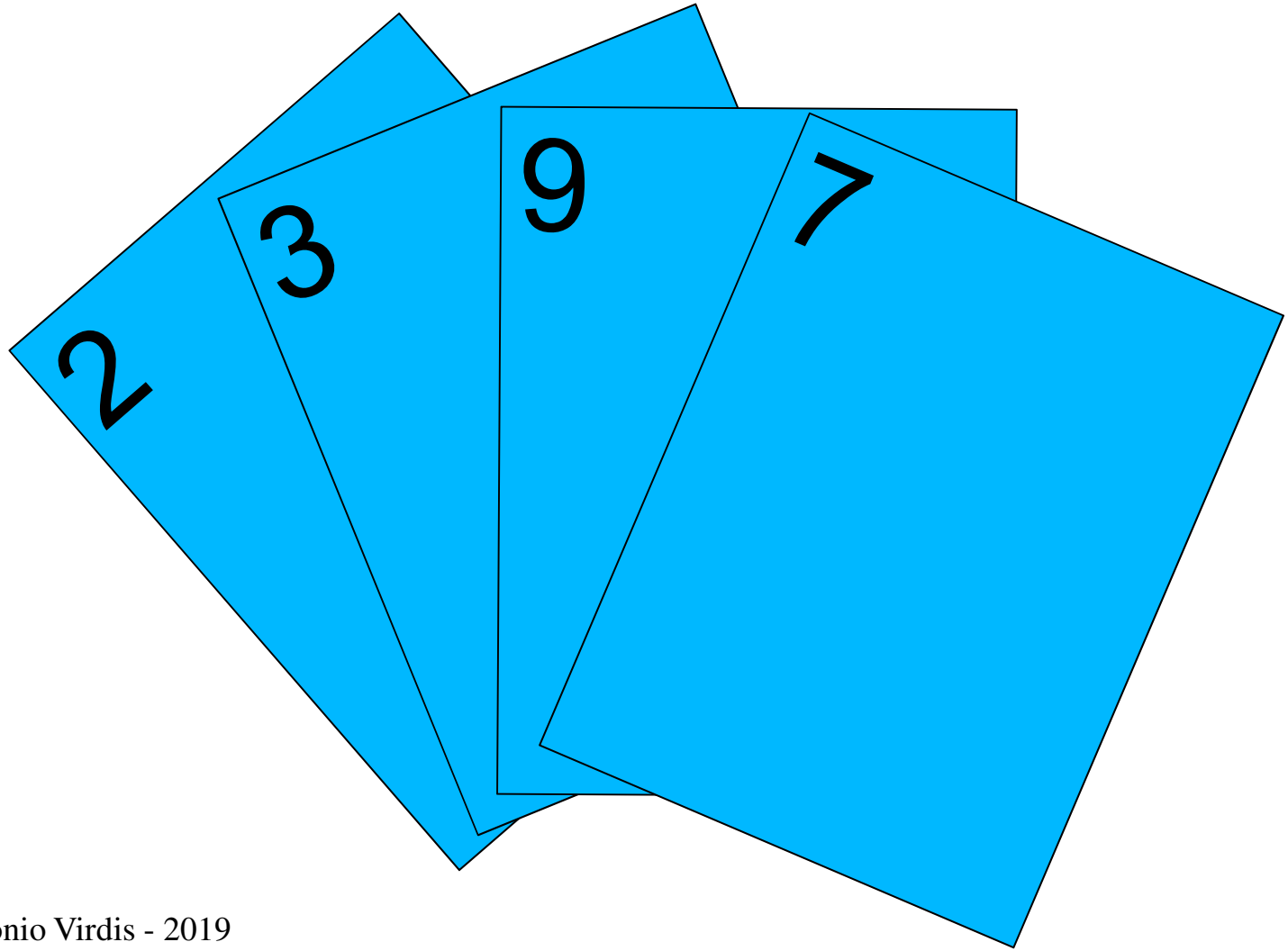
Ordinare Carte da Gioco



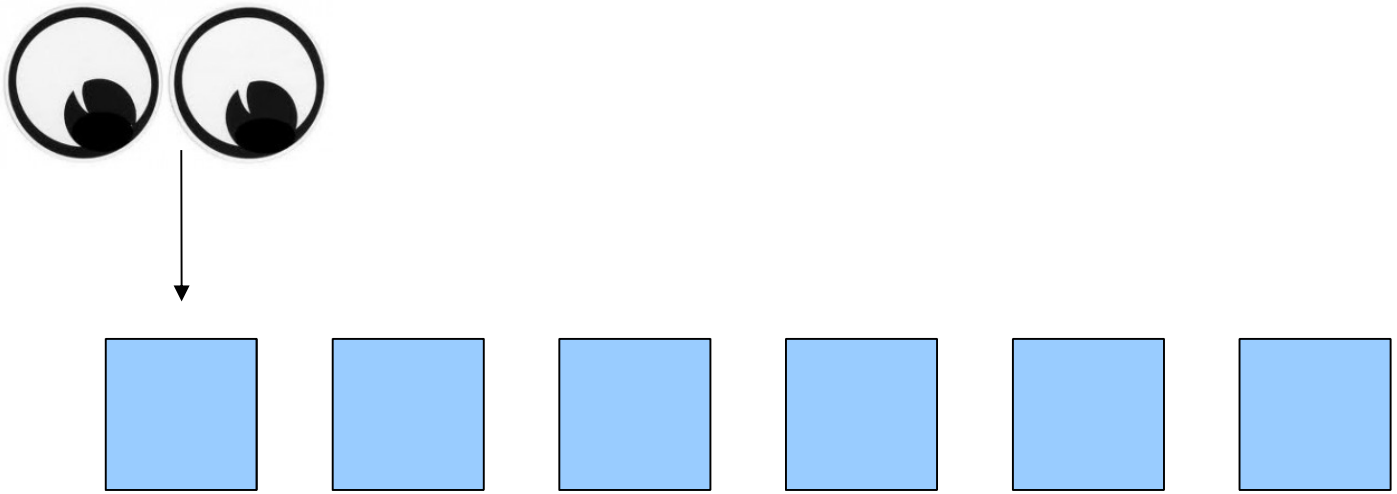
Ordinare Carte da Gioco



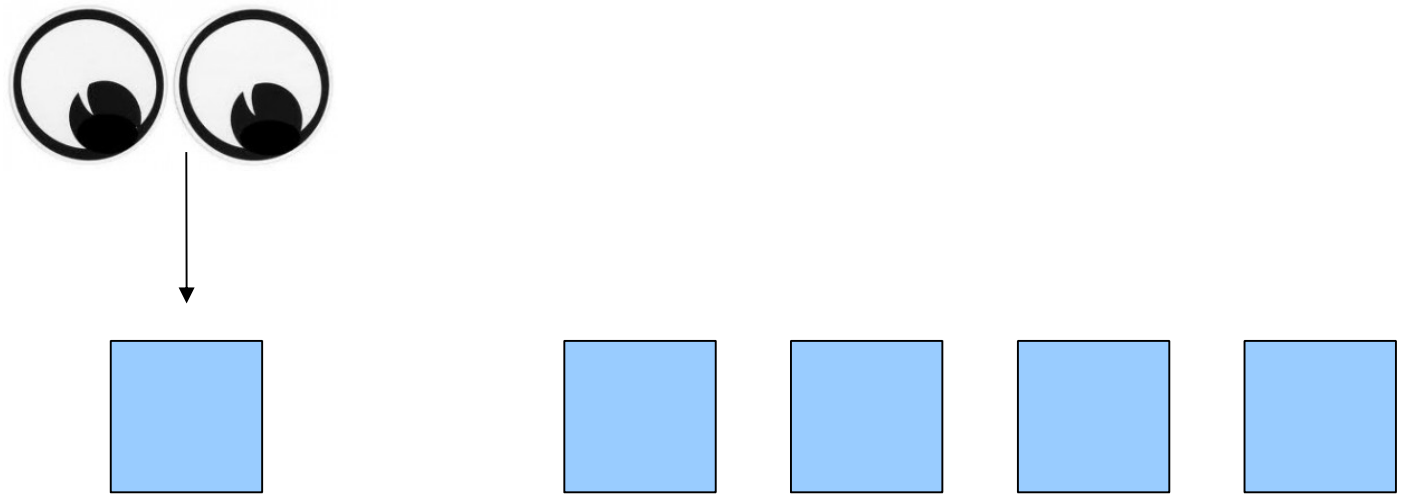
Ordinare Carte da Gioco

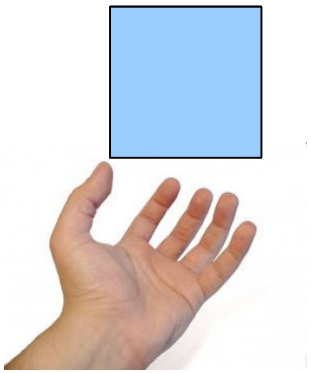


Insertion Sort: rappresentazione

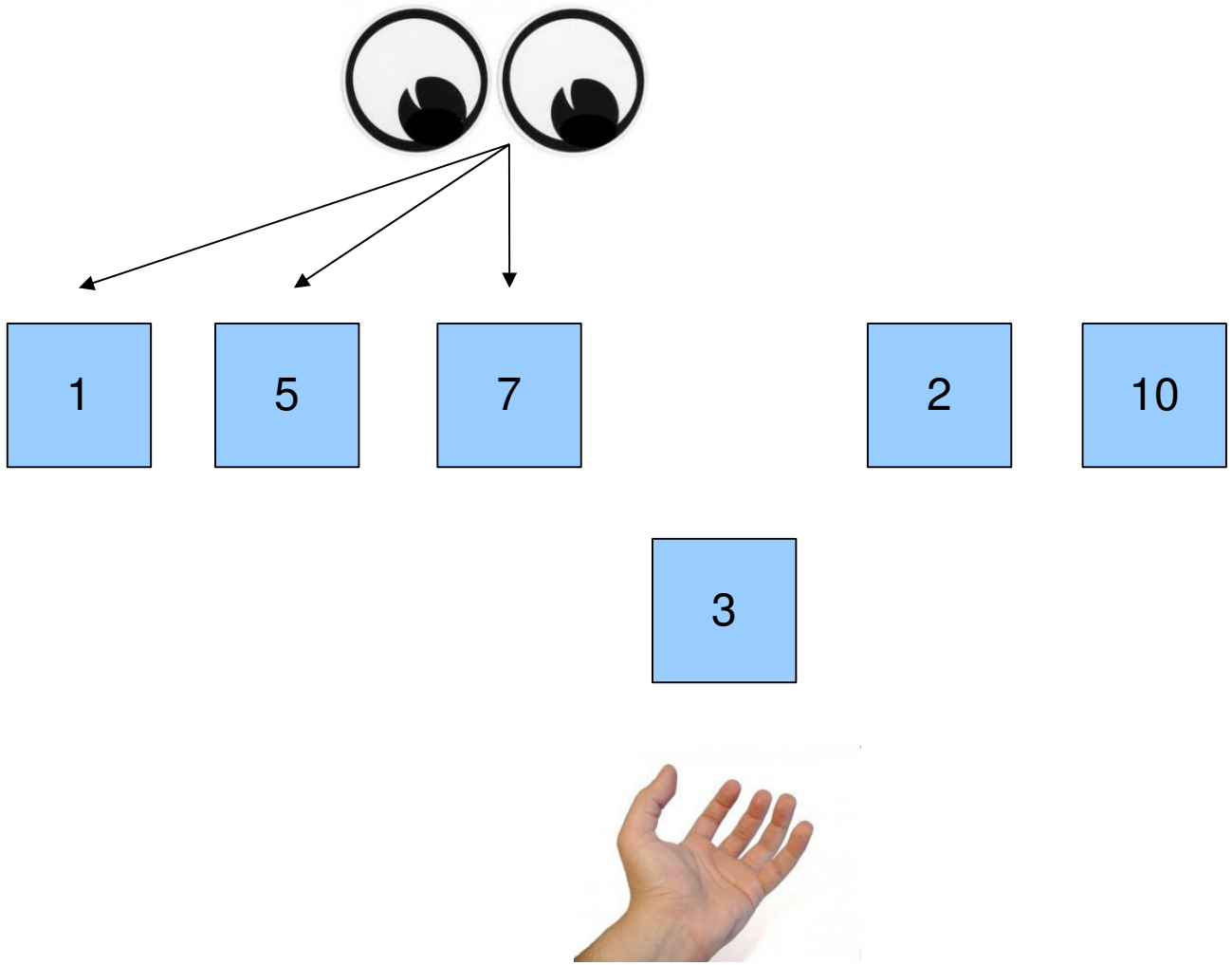


Insertion Sort: rappresentazione

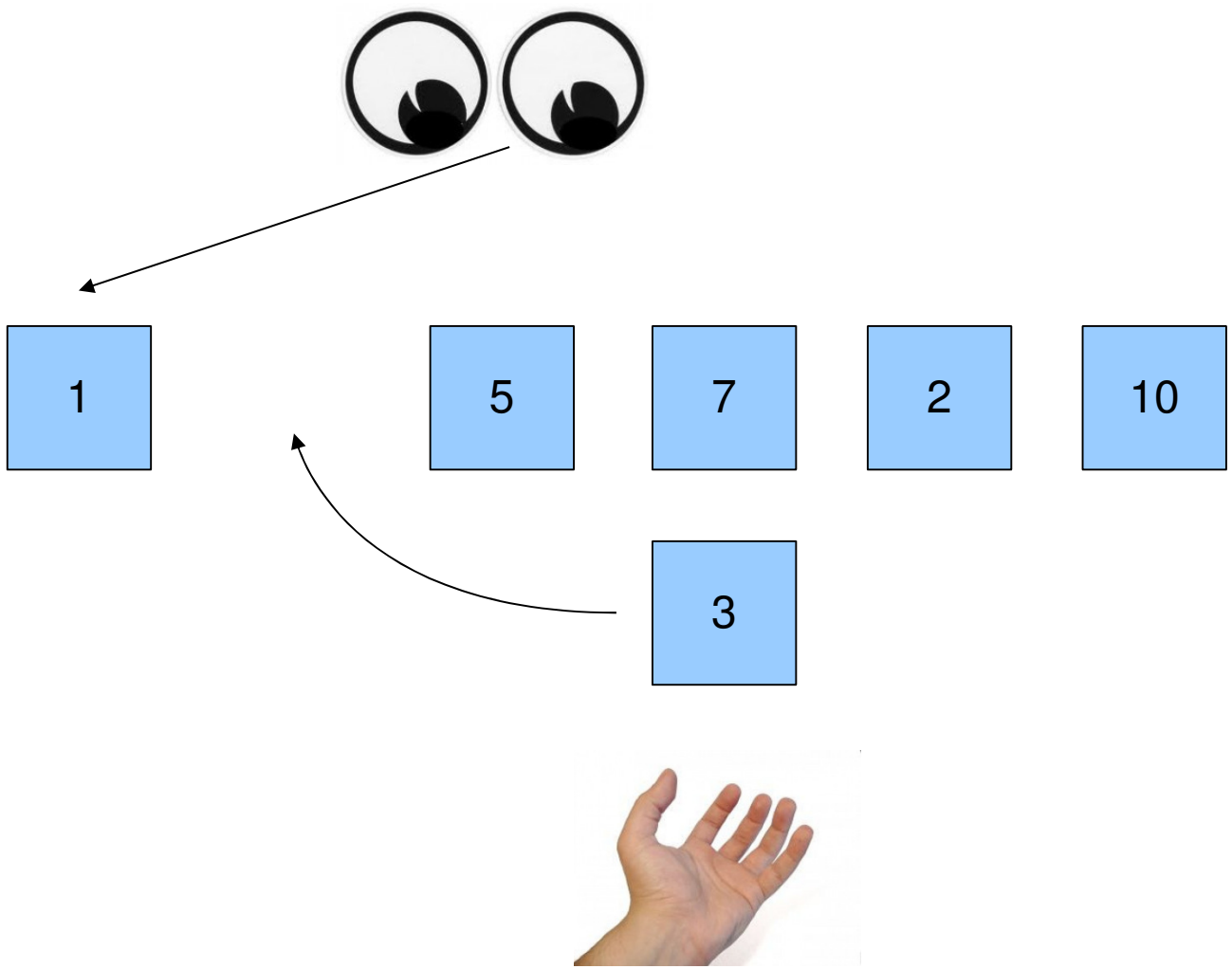




Insertion Sort: rappresentazione



Insertion Sort: rappresentazione



Sorting

```
1   void sortArray( int arr[] , int len )
2   {
3
4
5       for( PER OGNI ELEMENTO DELLA FILA )
6       {
7           // INIZIALIZZO MANO E OCCHIO
8
9
10          while( TROVO POSIZIONE CORRETTA )
11          {
12              // SPOSTO OGGETTO
13              // SPOSTO OCCHIO
14          }
15
16          // LIBERO MANO
17      }
18  }
```

Sorting

```
1   void sortArray( int arr[] , int len )
2   {
3       int mano = 0;
4       int occhio = 0;
5       for(           )
6       {
7
8
9
10          while(       )
11          {
12
13
14          }
15
16
17      }
18  }
```

Sorting

```
1   void sortArray( int arr[] , int len )
2   {
3       int mano = 0;
4       int occhio = 0;
5       for( int iter = 1 ; iter < len ; ++iter )
6       {
7           mano = arr[iter];
8           occhio = iter-1;
9
10          while( occhio >= 0 && arr[occhio] > mano )
11          {
12              arr[occhio+1] = arr[occhio];
13              --occhio;
14          }
15
16          arr[occhio+1] = mano;
17      }
18  }
```

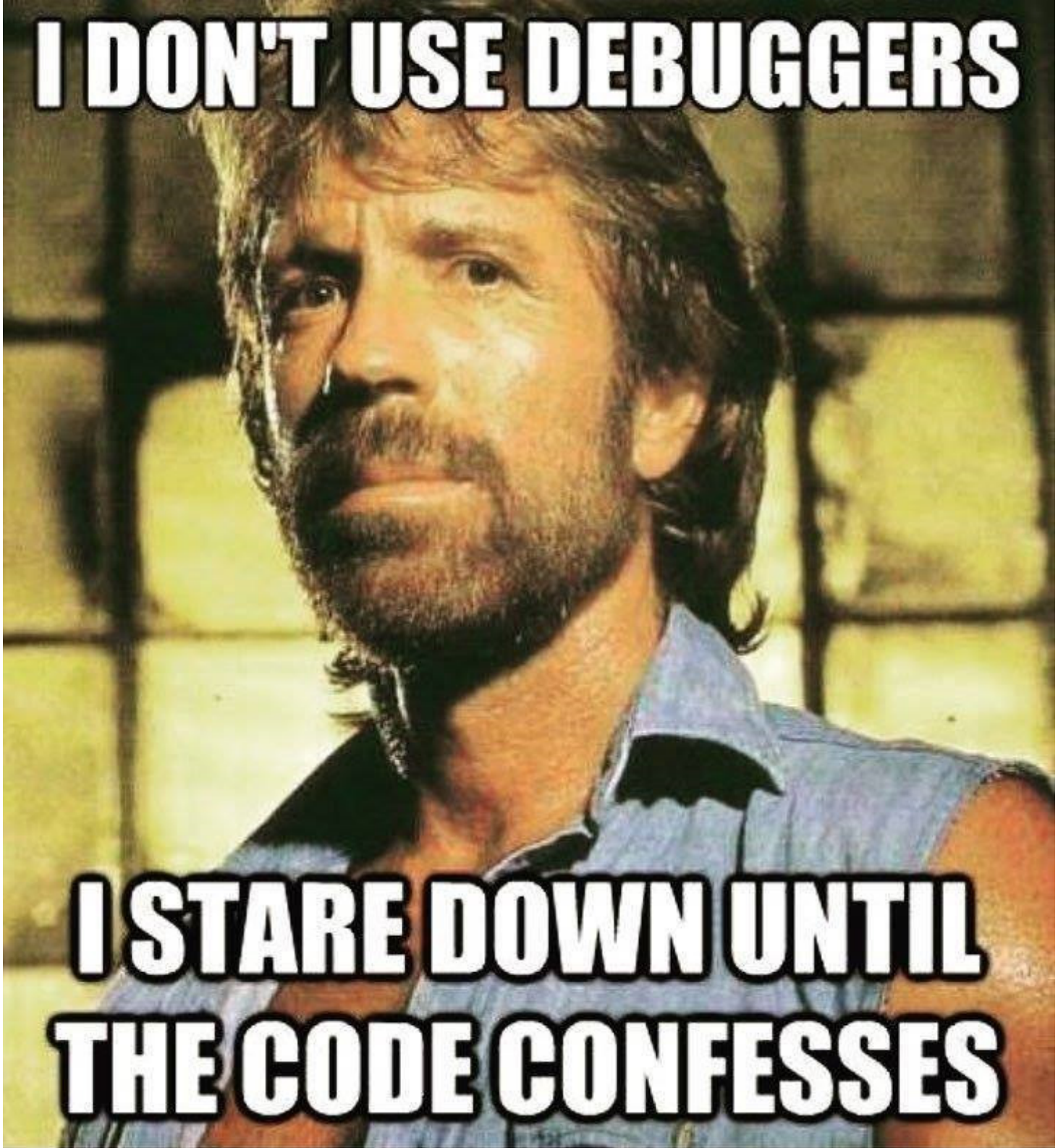
Debugging

“

If **debugging** is the process
of **removing** software bugs,

then **programming** must be the
process
of **putting** them in „

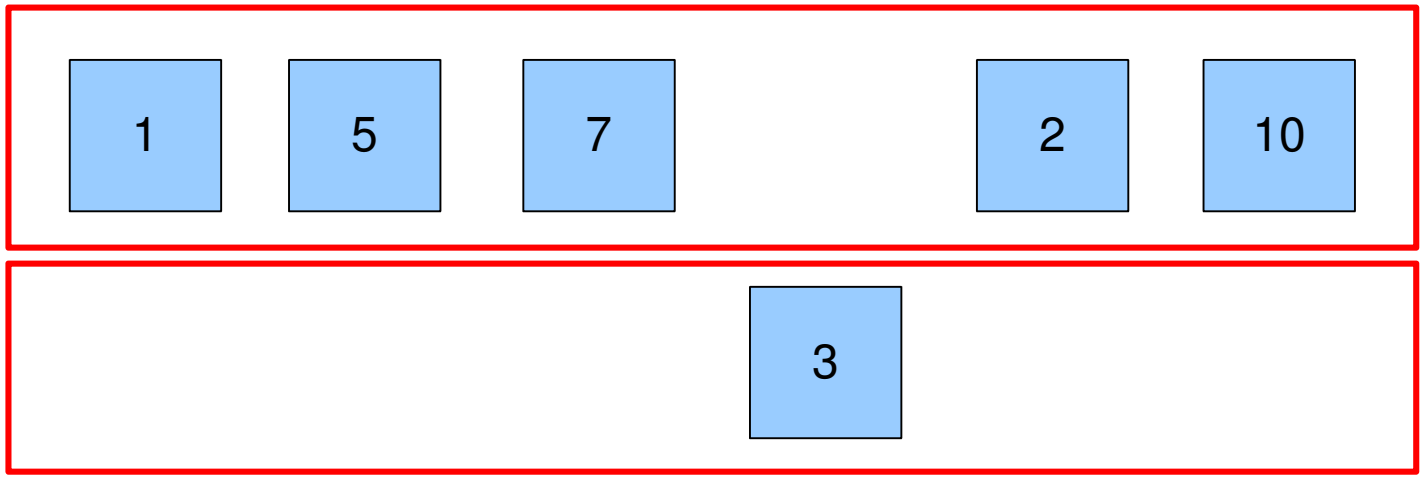
E. Dijkstra


I DON'T USE DEBUGGERS
I STARE DOWN UNTIL
THE CODE CONFESSES

Tecniche

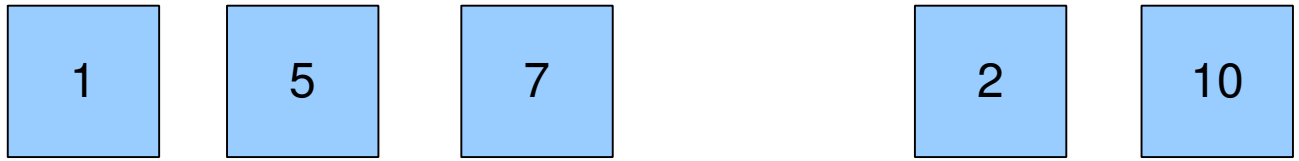
- Testo 
- Visuale
- “Debugger” (es **GDB**)
- Compilatore
- Analisi Memoria (**Valgrind**)

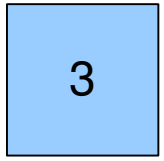
Debug Visuale



Debug Visuale

```
1   // stampa con buco
2   void stampaArray( int arr[] , int len , int buco)
3
4
5
6
7
8
9
10
11
12
13
14  // stampa "segno" in "posizione"
15  void stampaSegno( int posizione , int segno )
16
17
18
19
20
```



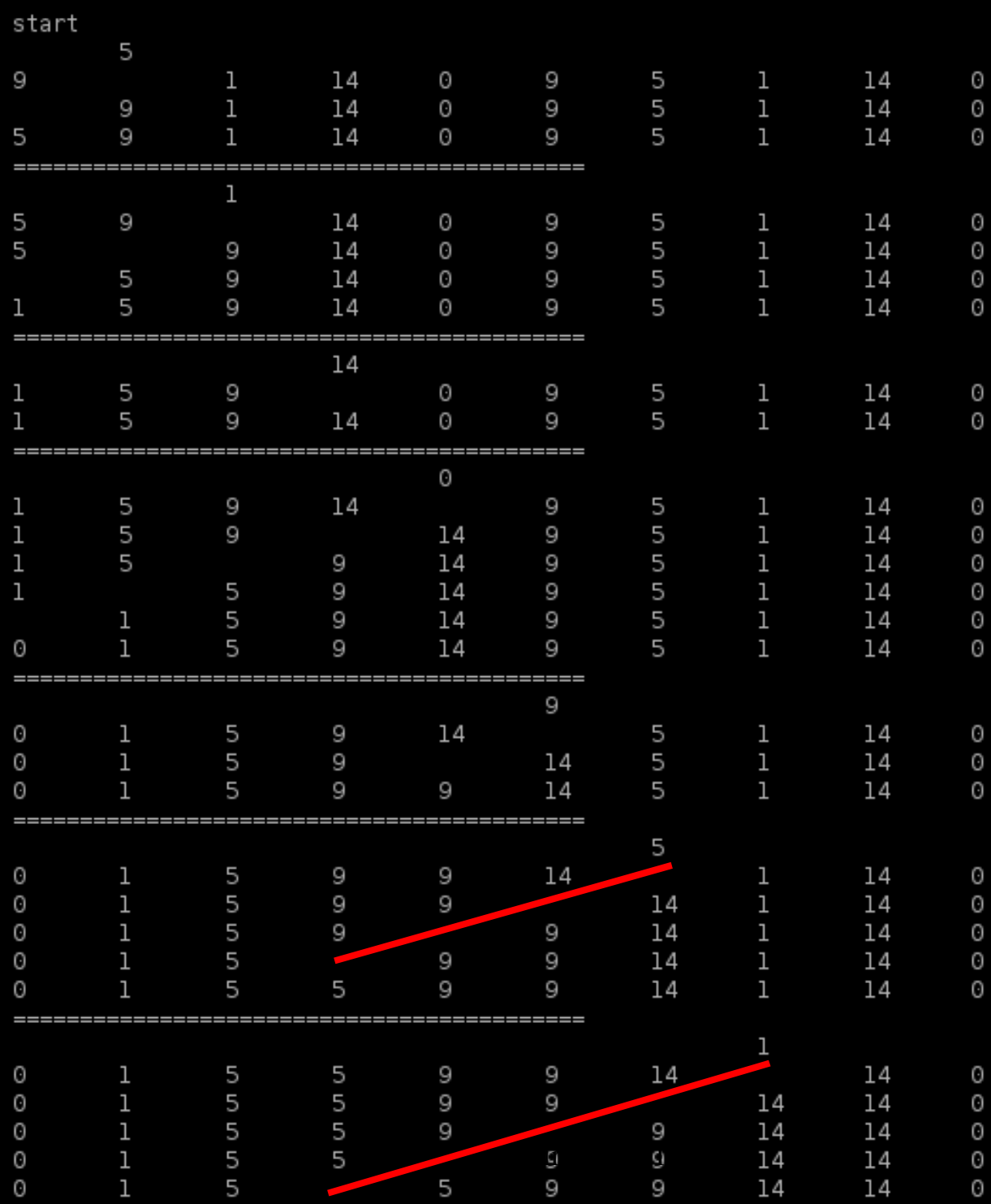


Debug Visuale

```
1   // stampa con buco
2   void stampaArray( int arr[] , int len , int buco)
3
4         // PER OGNI ELEMENTO
5
6         // SE SONO IN POSIZIONE buco, SALTO
7
8         // ALTRIMENTI STAMPO ELEMENTO
9
10
11
12
13
14  // stampa "segno" in "posizione"
15  void stampaSegno( int posizione , int segno )
16
17        // SALTO TUTTI GLI ELEMENTI FINO A posizione
18
19        // STAMPO IL SEGNO
20
```

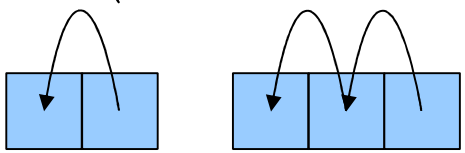
Debug Visuale

```
1   // stampa con buco
2   void stampaArray( int arr[] , int len , int buco)
3   {
4       for( int i=0 ; i < len ; ++i )
5       {
6           if(i==buco)
7               cout << "\t";
8           else
9               cout << arr[i] << "\t" ;
10      }
11      cout << endl;
12  }
13
14  // stampa "segno" in "posizione"
15  void stampaSegno( int posizione , int segno )
16  {
17      for( int i = 0 ; i < posizione ; ++i )
18          cout << "\t";
19      cout << segno << "\n";
20  }
```


start
5
9 1 14 0 9 5 1 14 0
9 1 14 0 9 5 1 14 0
5 9 1 14 0 9 5 1 14 0
1
5 9 14 0 9 5 1 14 0
5 9 14 0 9 5 1 14 0
5 9 14 0 9 5 1 14 0
1 5 9 14 0 9 5 1 14 0
14
1 5 9 0 9 5 1 14 0
1 5 9 14 0 9 5 1 14 0
0
1 5 9 14 9 5 1 14 0
1 5 9 14 9 5 1 14 0
1 5 9 14 9 5 1 14 0
1 5 9 14 9 5 1 14 0
1 5 9 14 9 5 1 14 0
0 1 5 9 14 9 5 1 14 0
9
0 1 5 9 14 5 1 14 0
0 1 5 9 14 5 1 14 0
0 1 5 9 9 14 5 1 14 0
5
0 1 5 9 9 14 1 14 0
0 1 5 9 9 14 1 14 0
0 1 5 9 9 14 1 14 0
0 1 5 9 9 14 1 14 0
0 1 5 5 9 9 14 1 14 0
1
0 1 5 5 9 9 14 14 0
0 1 5 5 9 9 14 14 0
0 1 5 5 9 9 14 14 0
0 1 5 5 9 9 14 14 0
0 1 5 5 9 9 14 14 0

Analisi InsertionSort

```
1   void sortArray( int arr[] , int len )
2   {
3
4
5       for( int iter = 1 ; iter < len ; ++iter )
6       {
7
8
9
10          while( occhio >= 0 && arr[occhio] > mano )
11          {
12
13
14          }
15
16
17      }
18  }
```



...

$$\frac{n(n-1)}{2}$$

$$\Theta(n^2)$$

Esempio Worst Case

- Input di worst case?

10 , 9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1

```
1   void sortArray( int arr[] , int l )
2   {
3       // init total e counter
4
5       for( int iter = 1 ; iter < len ; ++iter )
6       {
7
8           counter = 0;
9
10          while( occhio >= 0 && arr[occhio] > mano )
11          {
12              counter++;
13
14          }
15
16          total += counter;
17      }
18  }
```

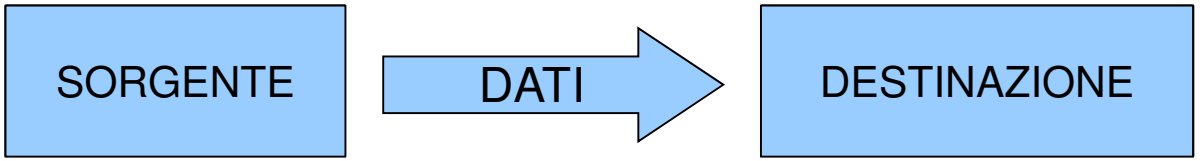
```
start
        9
10              8       7       6       5       4       3       2       1
        10      8       7       6       5       4       3       2       1
9       10      8       7       6       5       4       3       2       1
=========================================
================  1  ====================
                8
9       10              7       6       5       4       3       2       1
9               10      7       6       5       4       3       2       1
        9       10      7       6       5       4       3       2       1
8       9       10      7       6       5       4       3       2       1
=========================================
================  2  ====================
                        7
8       9       10              6       5       4       3       2       1
8       9               10      6       5       4       3       2       1
8               9       10      6       5       4       3       2       1
        8       9       10      6       5       4       3       2       1
7       8       9       10      6       5       4       3       2       1
=========================================
================  3  ====================
                                6
7       8       9       10              5       4       3       2       1
7       8       9               10      5       4       3       2       1
7       8               9       10      5       4       3       2       1
7               8       9       10      5       4       3       2       1
        7       8       9       10      5       4       3       2       1
6       7       8       9       10      5       4       3       2       1
=========================================
================  4  ====================
                                        5
6       7       8       9       10              4       3       2       1
6       7       8       9               10      4       3       2       1
6       7       8               9       10      4       3       2       1
6       7               8       9       10      4       3       2       1
6               7       8       9       10      4       3       2       1
        6       7       8       9       10      4       3       2       1
5       6       7       8       9       10      4       3       2       1
=========================================
================  5  ====================
```

Memoria dinamica

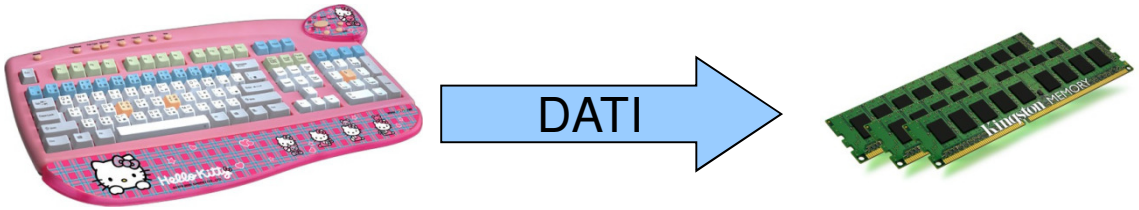
- Quantità di dati **NON** nota a tempo di compilazione

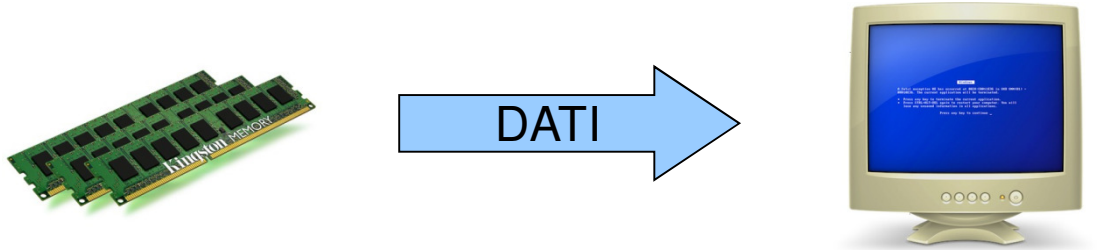
- Quantità di dati **VARIABILE** durante l'esecuzione

Lettura Input



`std::cin`

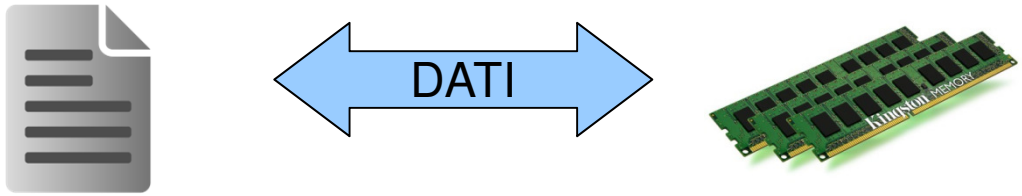




`std::cout`

`std::ifstream`

`std::ofstream`



Redirezione DA File

<

```
./stlSort < reqFile
```

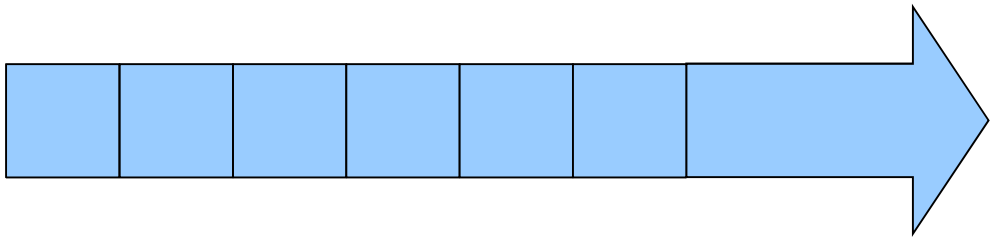
Lettura Input

```
1
2
3       leggiInput( )
4
5
6         // 1) LEGGO PRIMO VALORE (numero elementi)
7
8
9         // 2) ALLOCAZIONE MEMORIA
10
11
12        // 3) LETTURA CARATTERE PER CARATTERE
13
14
15
16
17
18
```

Lettura Input

```
1
2   int * leggiInput( )
3   {
4
5       cin >> len;
6
7       int * arr = new int[len];
8
9
10      for( int i = 0 ; i < len ; ++i )
11          cin >> arr[i];
12
13
14      return arr;
15  }
16
17
18
```

Vettore



- Struttura dati di dimensione estendibile
- Accesso efficiente
- Algoritmi
- Magari già pronta?!?

Standard Template Library (STL)

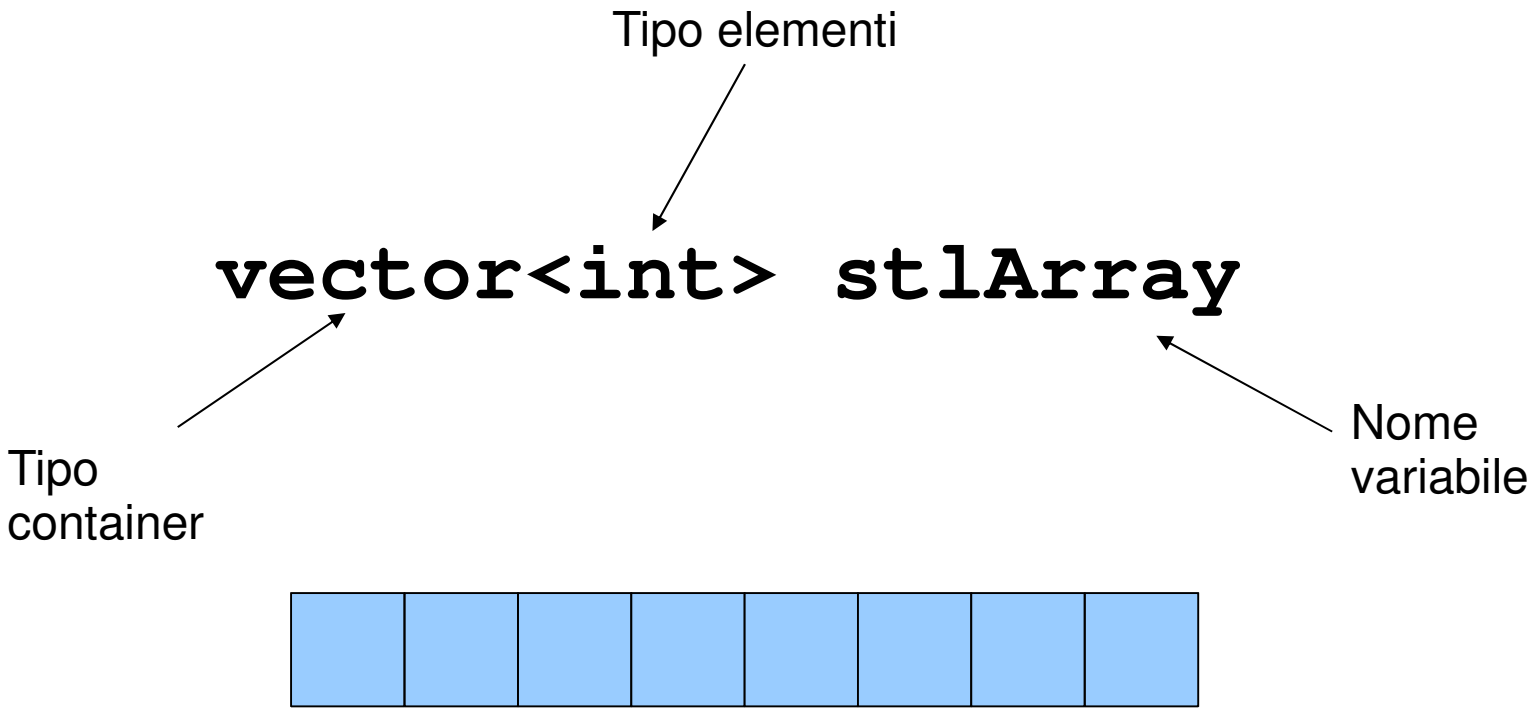
containers

iterators

algorithms

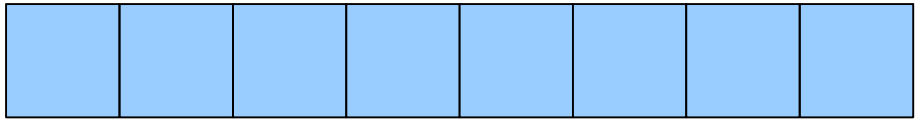
Documentazione:
www.cplusplus.com/reference/stl/

Uso Vector


Tipo elementi
vector<int> stlArray
Tipo
container
Nome
variabile

Uso Vector

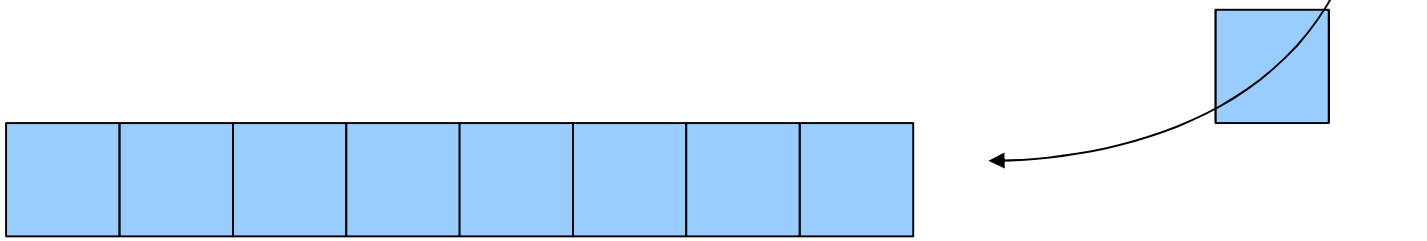
```
vector<int> stlArray
```



Uso Vector

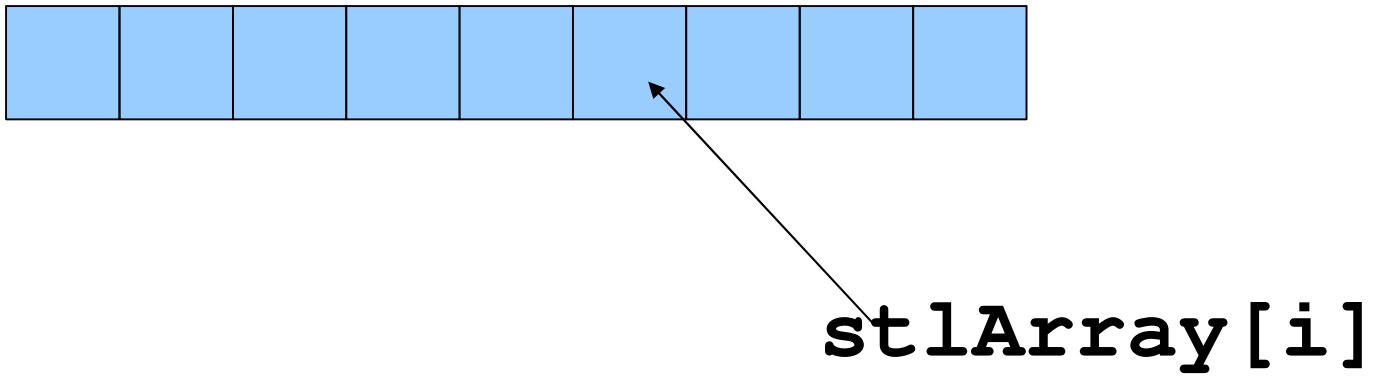
```
vector<int> stlArray
```

```
stlArray.push_back(val)
```



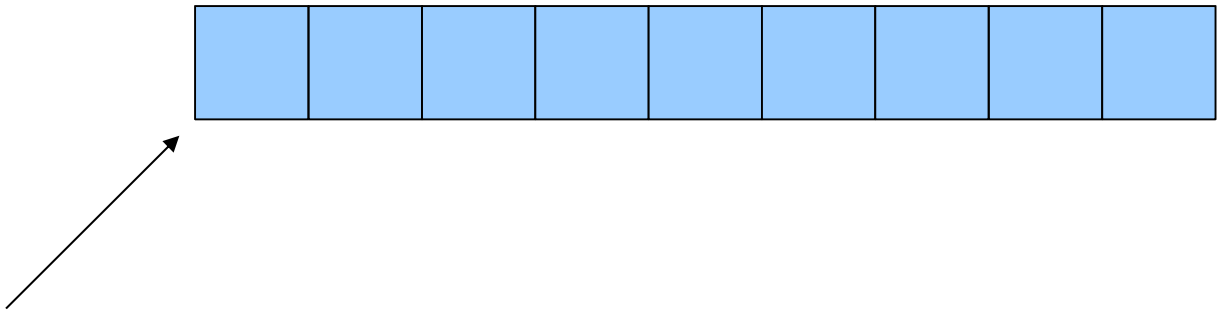
Uso Vector

```
vector<int> stlArray
```



Uso Vector

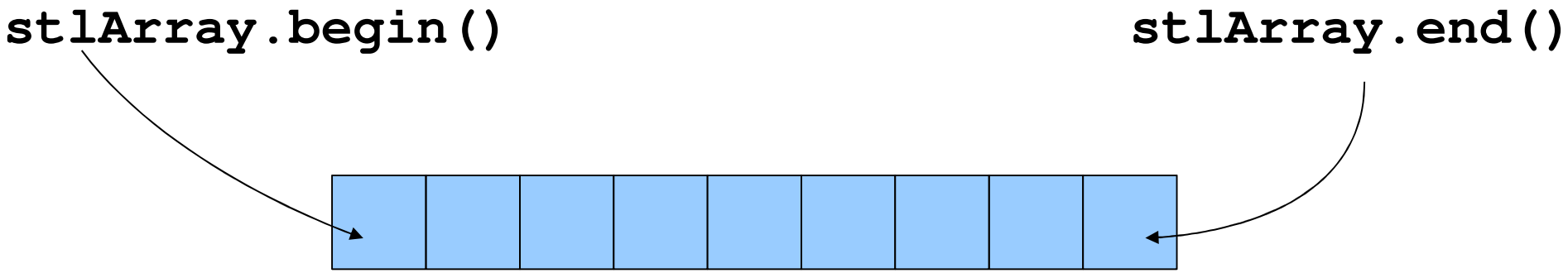
```
vector<int> stlArray
```



```
&stlArray[0]
```

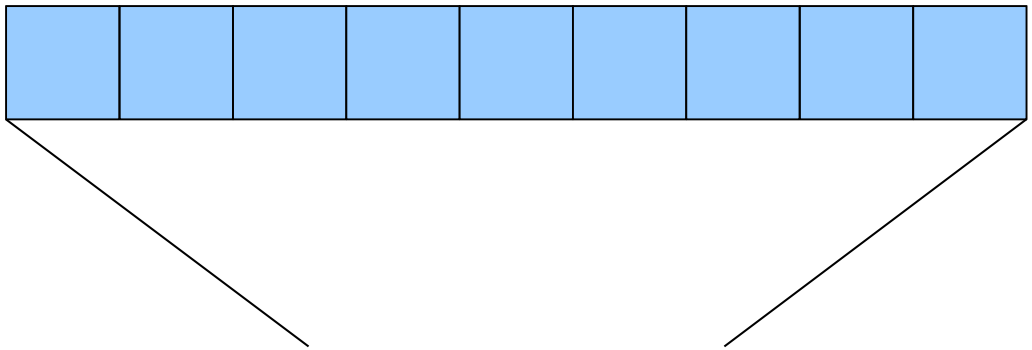
Uso Vector

```
vector<int> stlArray
```



Uso Vector

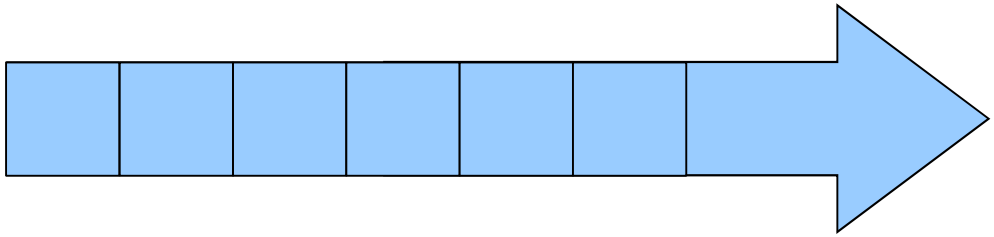
```
vector<int> stlArray
```

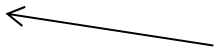


```
stlArray.size()
```

Uso Vector

- Dinamico
  - Allocazione dinamica dimensione



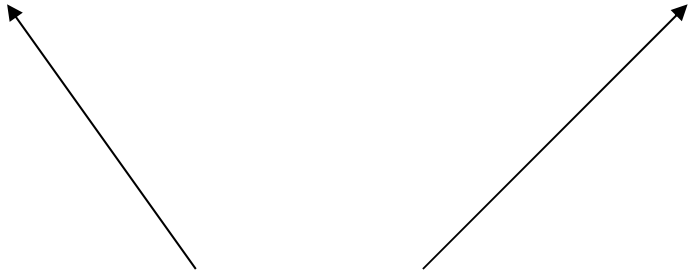
- Contiguo
  - Accesso Random con costo costante
  - Gestione array-like  **(con prudenza)**

File ⇒ Vector

```
1   #include <vector>
2
3   void leggiInput( std::vector<int> & arr )
4   {
5
6       cin >> len;
7
8       int val;
9       for( int i = 0 ; i < len ; ++i )
10      {
11          cin >> val;
12          arr.push_back(val);
13      }
14
15      return;
16  }
17
18
```

Sort STL

```
sort( stlArray.begin(),stlArray.end() );
```



$$\Theta(n \log n)$$

Qualche Test

Insertion Sort

vs

STL Sort

$\Theta(n^2)$

$\Theta(n \log n)$

```
time ./stlSort
```

Qualche Test

- Casi limite
  - Tutti uguali
  - Già ordinato
  - Ordine inverso
- Valori random
  - srand(seed) rand()%maxVal
- Comando time
  - time nomeEseguibile

Come Esercitarsi

- Input: input.txt Output: output.txt

LETTURA

```
cin >> valore;
```

INPUT

```
./eseguibile < input.txt
```

GENEZIONE OUTPUT

```
cout << uscita;
```

VERIFICA

```
./eseguibile < input.txt | diff - output.txt
```

Esercizio

Input: `input.txt`

```
3
1
9
15
```

Output: `output.txt`

```
25
135
yes
```

Input

- Il primo carattere indica il numero di valori da leggere
- Un valore per riga

Output

- Somma dei valori
- Prodotto dei valori
- I valori sono positivi? Rispondere yes o no