

Algoritmi e Strutture Dati

Lezione 4

www.iet.unipi.it/a.virdis

Antonio Virdis

antonio.virdis@unipi.it

Sommario

- (recap) Comandi base Unix
- Ordinamento: Insertion Sort
- Standard Template Library
- Debug

esperimenti

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

(recap) Comandi Base Unix

- Navigare tra le cartelle
 - `ls`
 - `pwd`
 - `cd cartella`
 - `cd ..`

(recap) Comandi Base Unix

- Creare/spostare/copiare/cancellare file e cartelle
 - `mkdir cartella`
 - `touch file`
 - `cp/mv sorgente destinazione`
 - `rm file`
 - `rm -R cartella`

(recap) Comandi Base Unix (2)

- Compilare un singolo file C++
 - `g++ [opzioni] -o eseguibile sorgente.cpp`
- Eseguire un programma
 - `./eseguibile`

Ordinamento



Antonio Virdis - 2022

Stampa

```
/*Scrivere un programma che:
- possa memorizzare 10 interi
- memorizzi 10 interi
- usi una funzione per stampare i 10 interi */
int main()
{
}
}
```


Stampa

```
/*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;  
  
}
```

Stampa

```
/*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

```
// 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
2
3
4 void stampaArray( int arr[] , int len )
5
6
7
8
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 }
```

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 }
```

Esercizio 0

1. creare nella vostra home un file `esercizio0.cpp`
2. creare nella vostra home una cartella `lezione4`
3. spostare il file `esercizio0.cpp` dentro la cartella `lezione4`
4. implementare il codice delle slide precedente
5. compilare ed eseguire
6. copiare una copia del file `esercizio0.cpp`
7. ripetere il passo 5 per il file copiato
8. cancellare il file copiato

Nomi delle Variabili



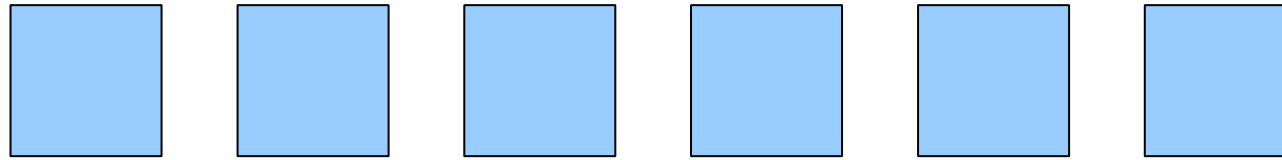
Antonio Virdis - 2022

prova pippo

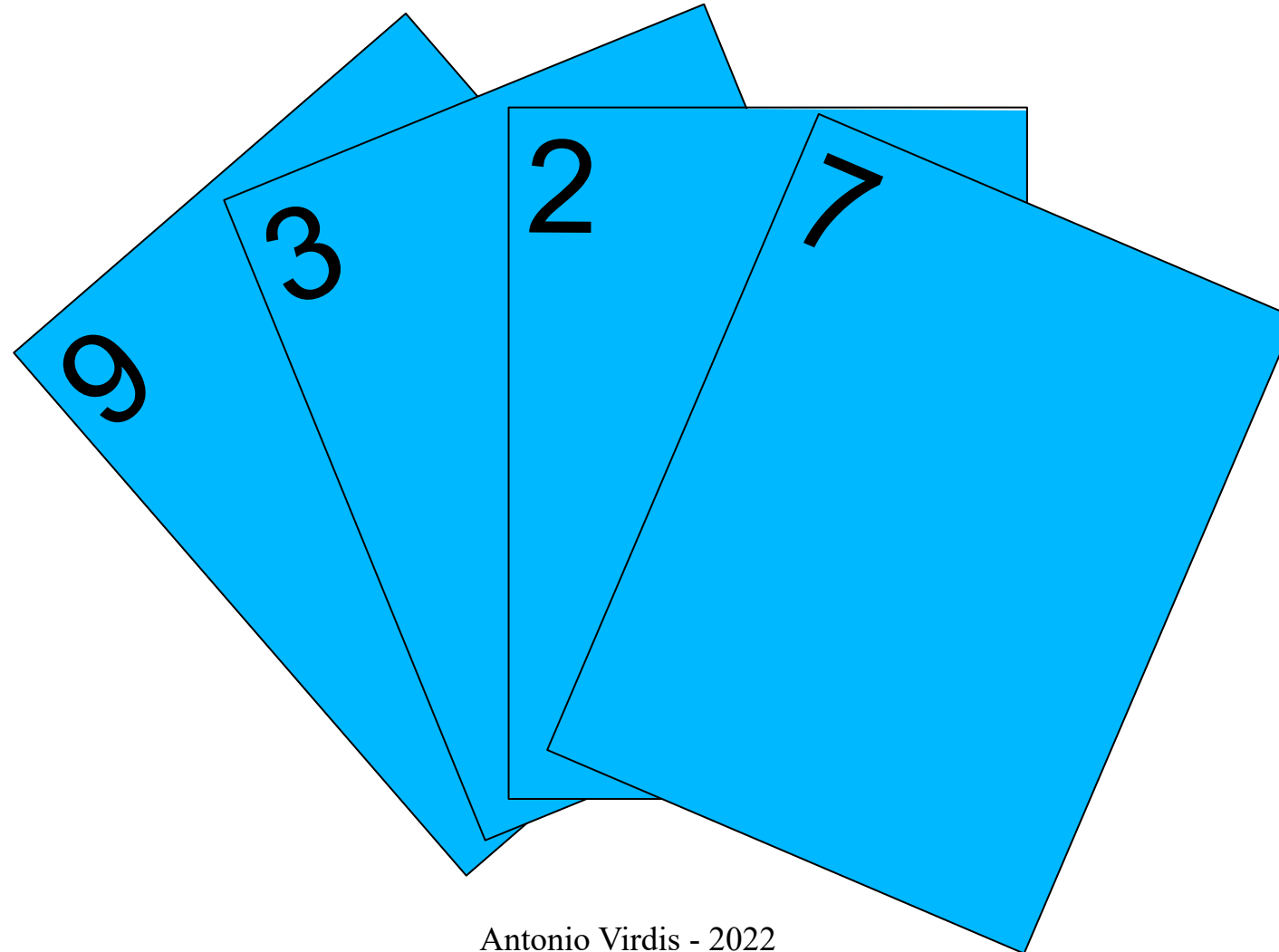
```
1  
2  
3  
4  
5  
6  
7 int main( ){const int a = 10;  
8 int pippo[a] = { 9,5,1,14,0,9,5,1,14,0 };  
9 prova(pippo,a);  
10 return 1;}  
11  
12  
13  
14  
15  
16  
17  
18
```

?

Insertion Sort: rappresentazione

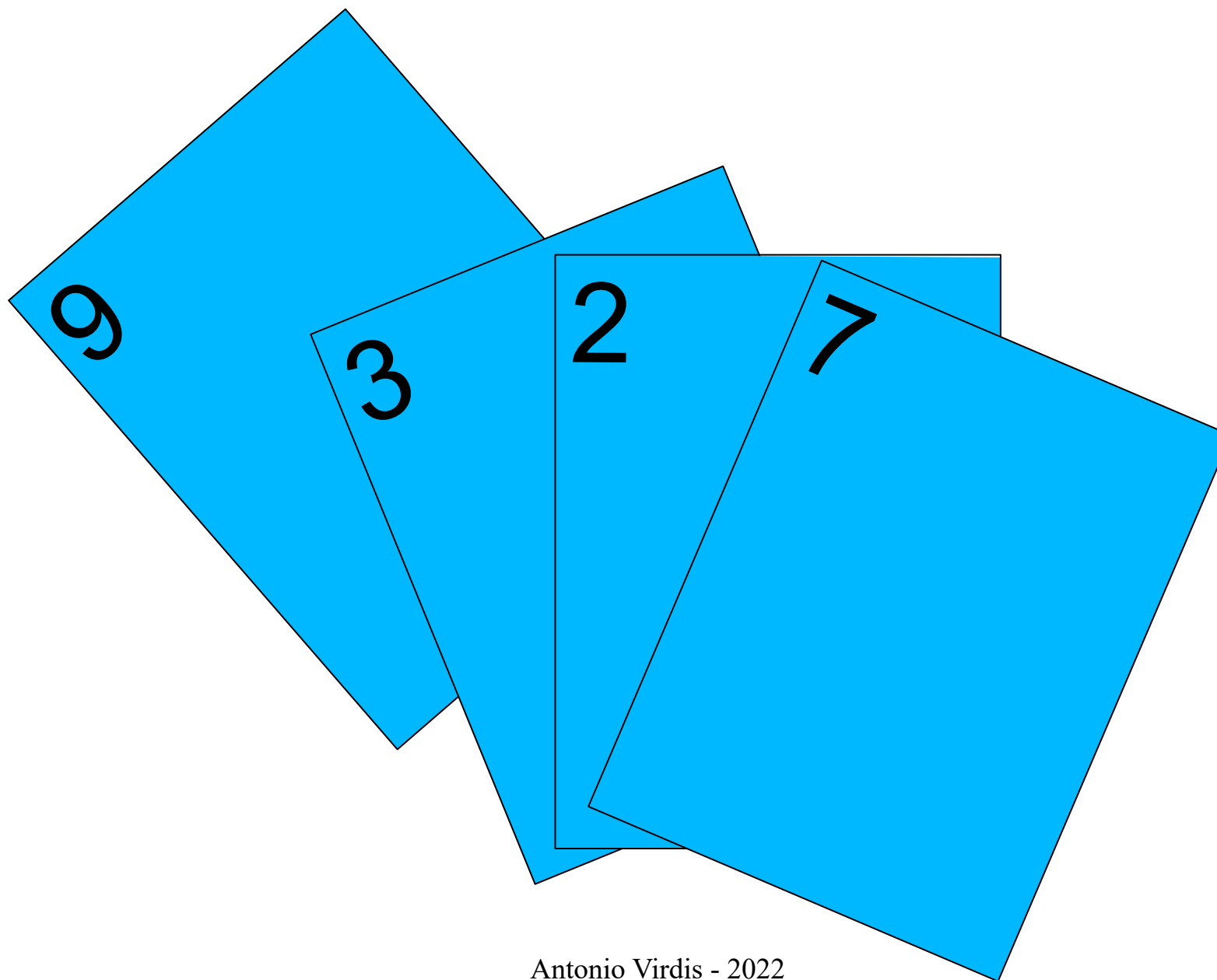


Ordinare Carte da Gioco



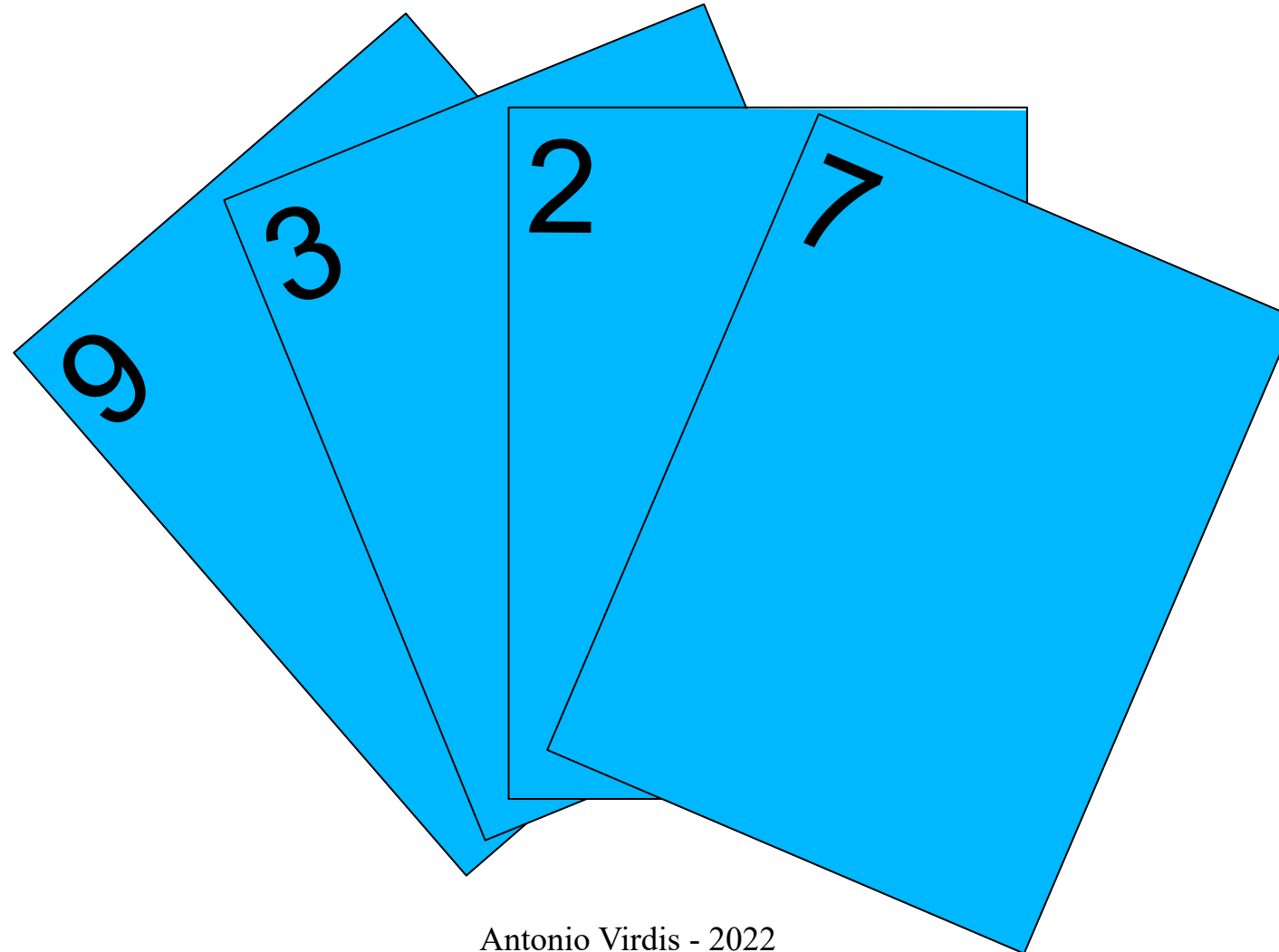
Antonio Virdis - 2022

Ordinare Carte da Gioco



Antonio Virdis - 2022

Ordinare Carte da Gioco



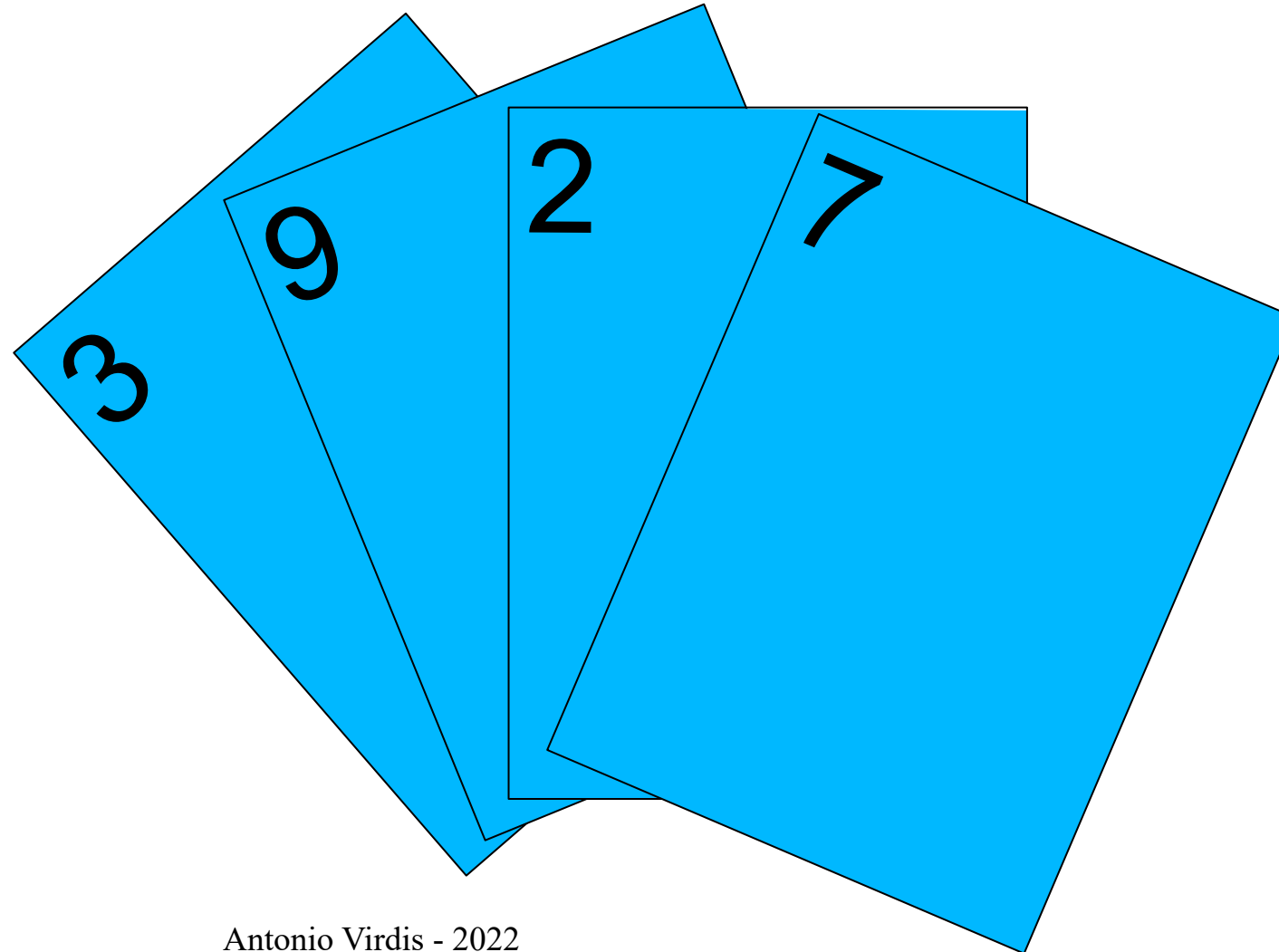
Antonio Virdis - 2022

Ordinary Carte da Gioco



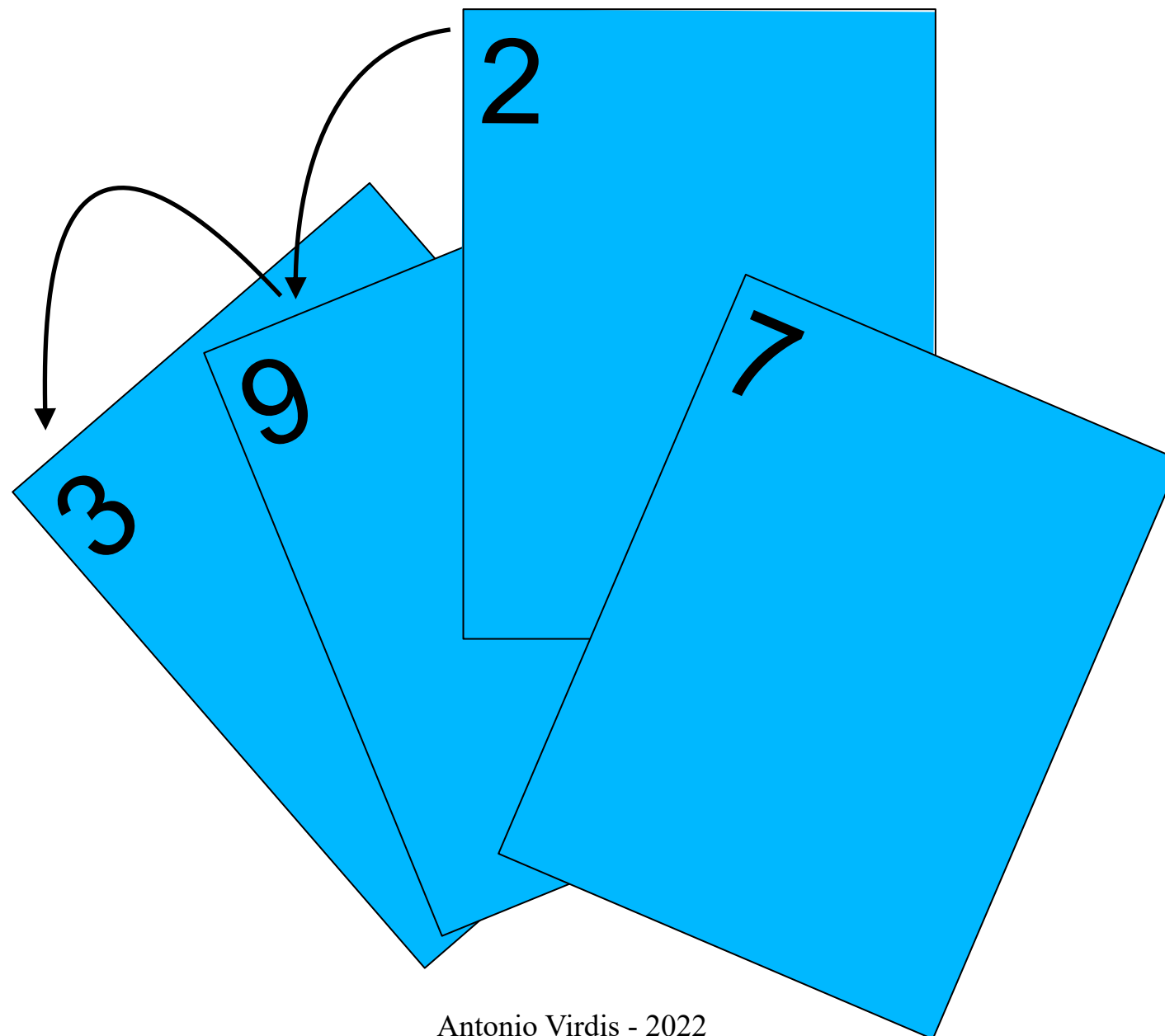
Antonio Virdis - 2022

Ordinare Carte da Gioco



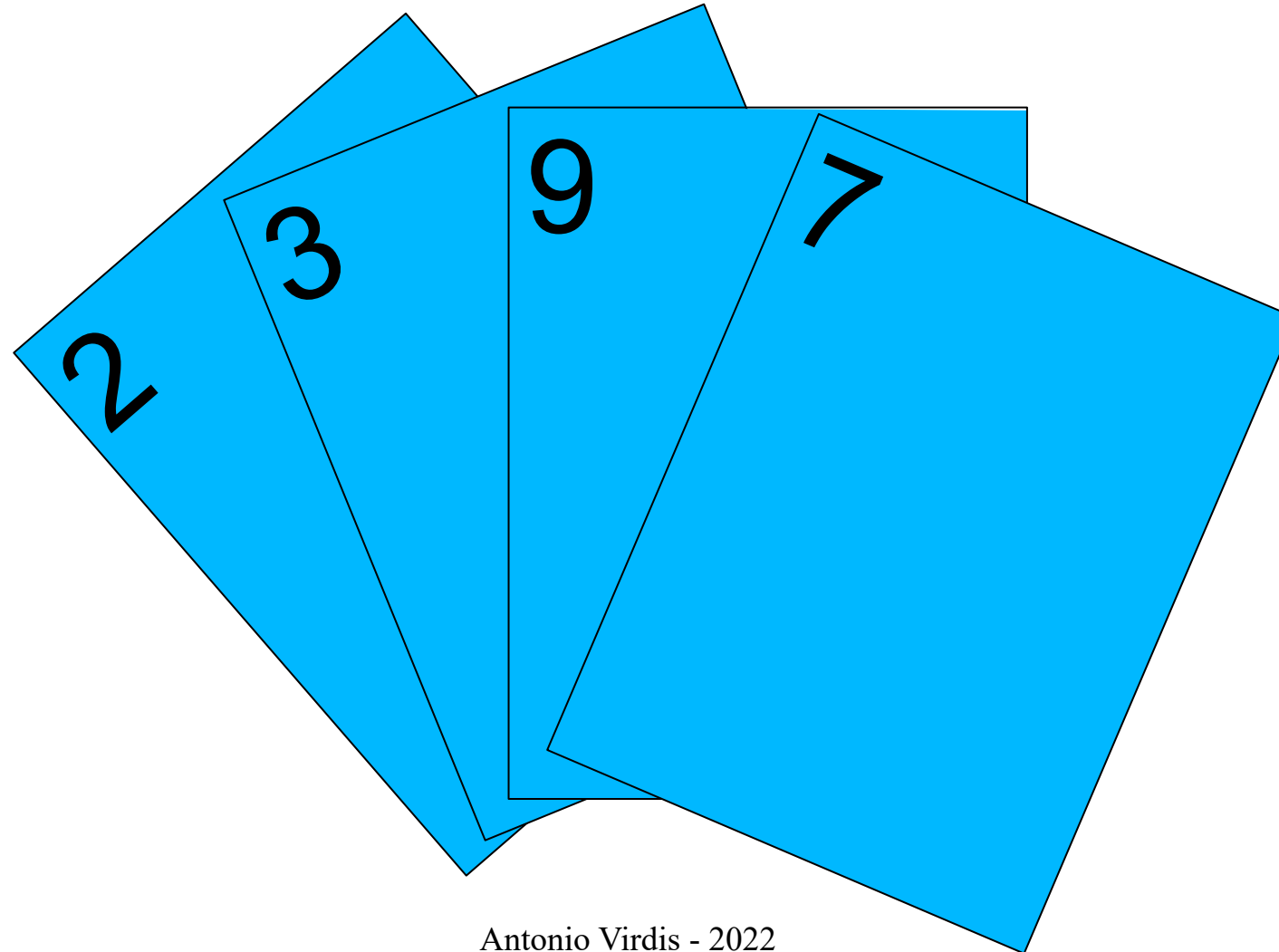
Antonio Virdis - 2022

Ordinare Carte da Gioco



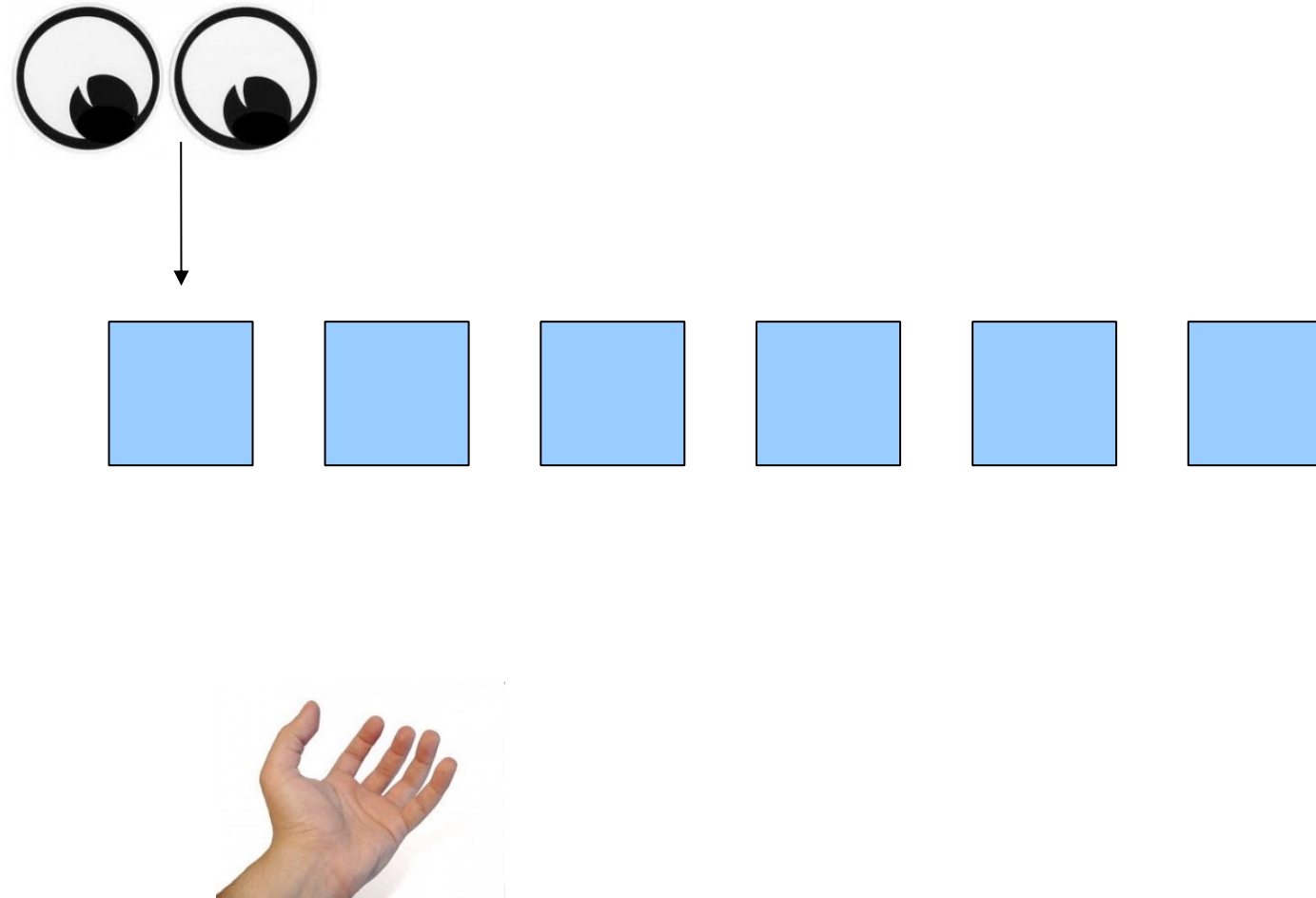
Antonio Virdis - 2022

Ordinare Carte da Gioco

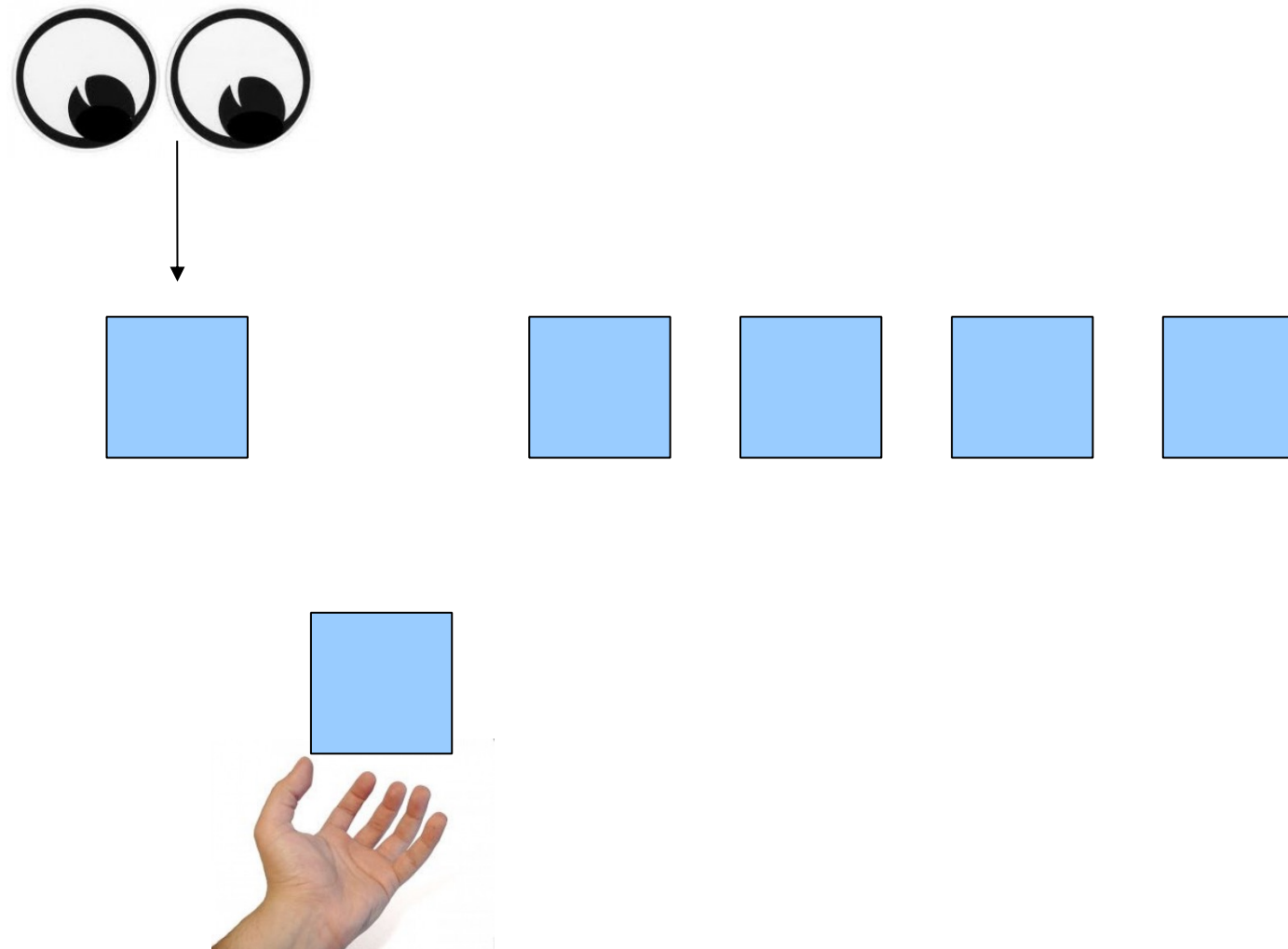


Antonio Virdis - 2022

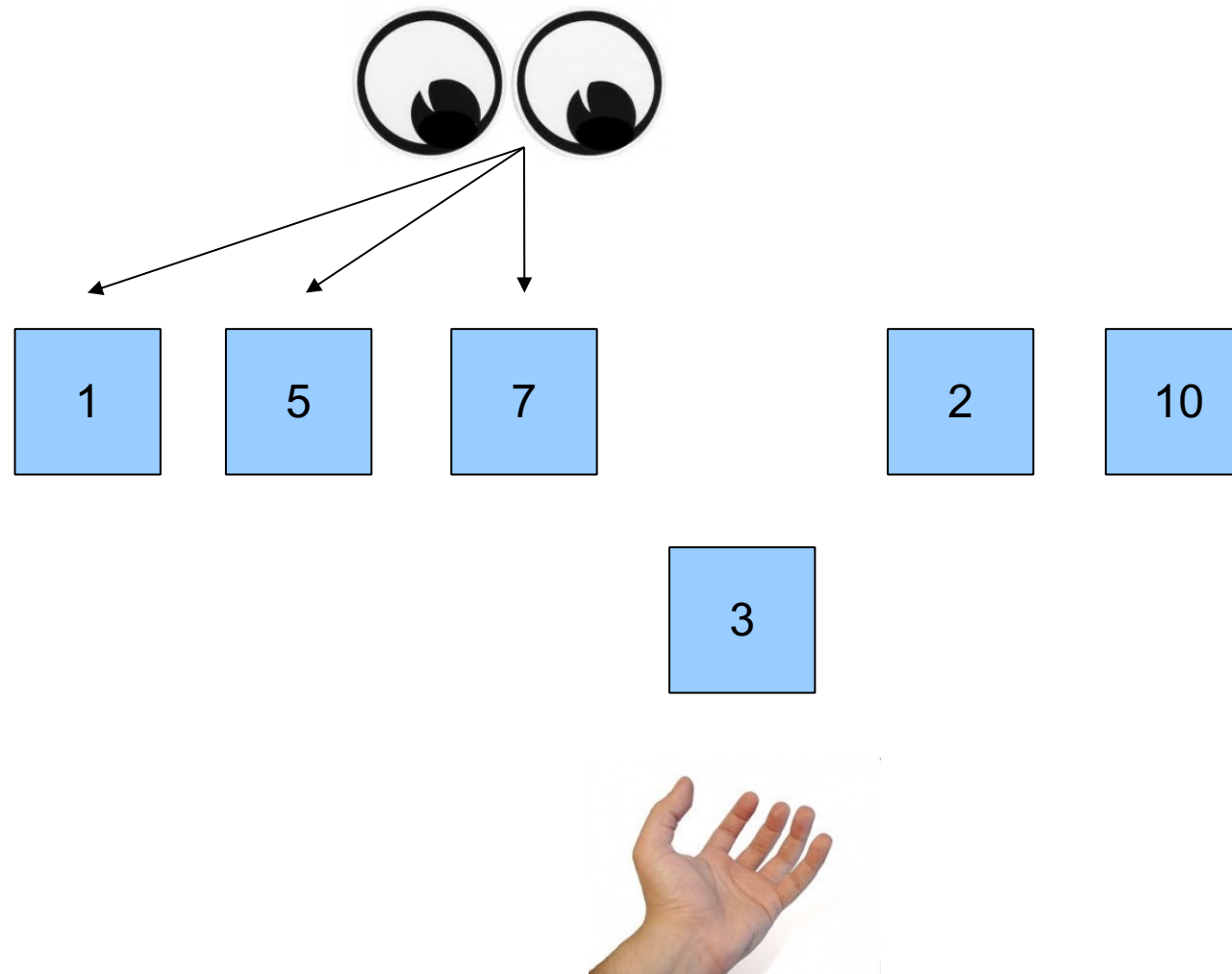
Insertion Sort: rappresentazione



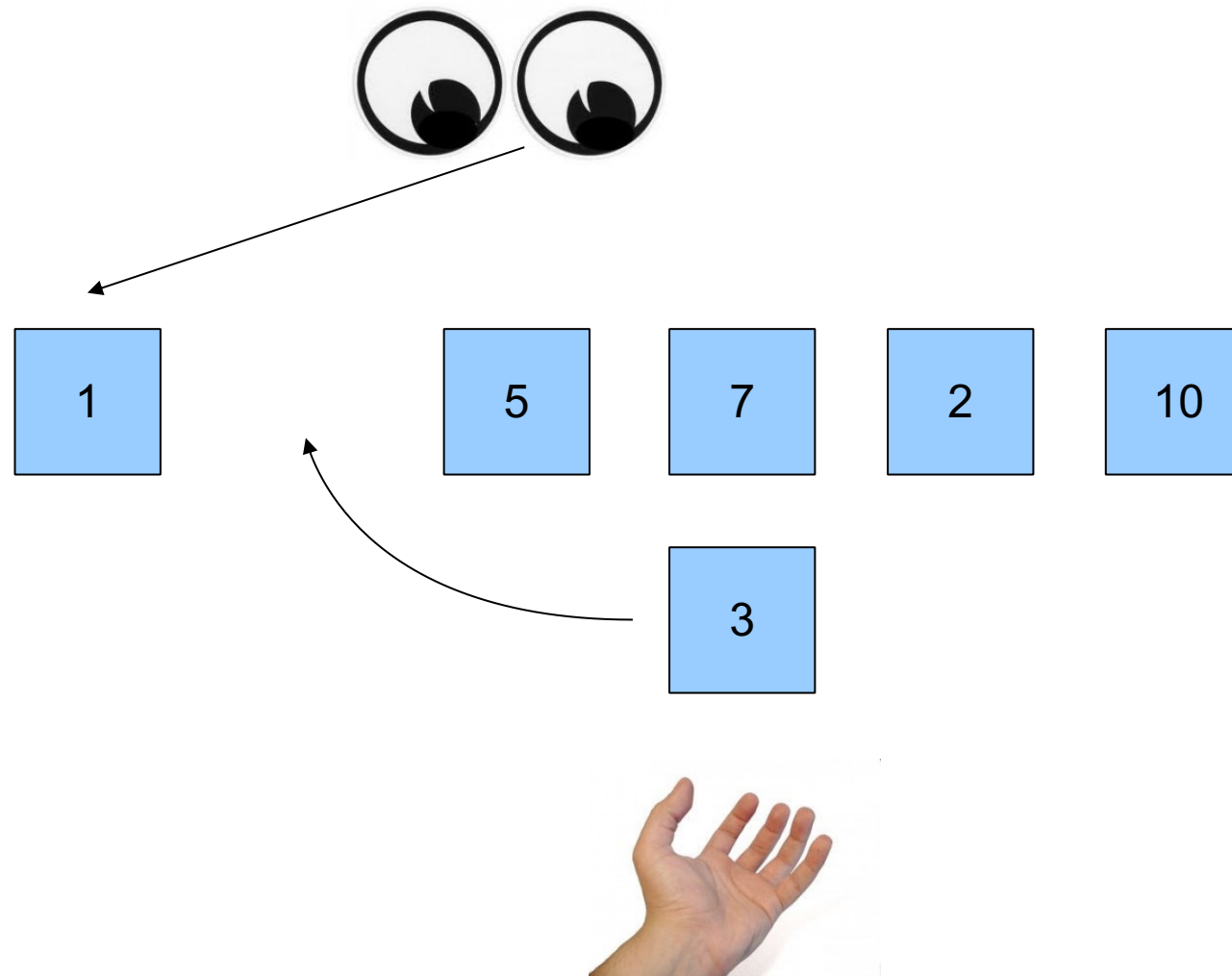
Insertion Sort: rappresentazione



Insertion Sort: rappresentazione



Insertion Sort: rappresentazione



Sorting

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

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
11
12
13
14
15
16
17     }
18 }
```

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(          )
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    }
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

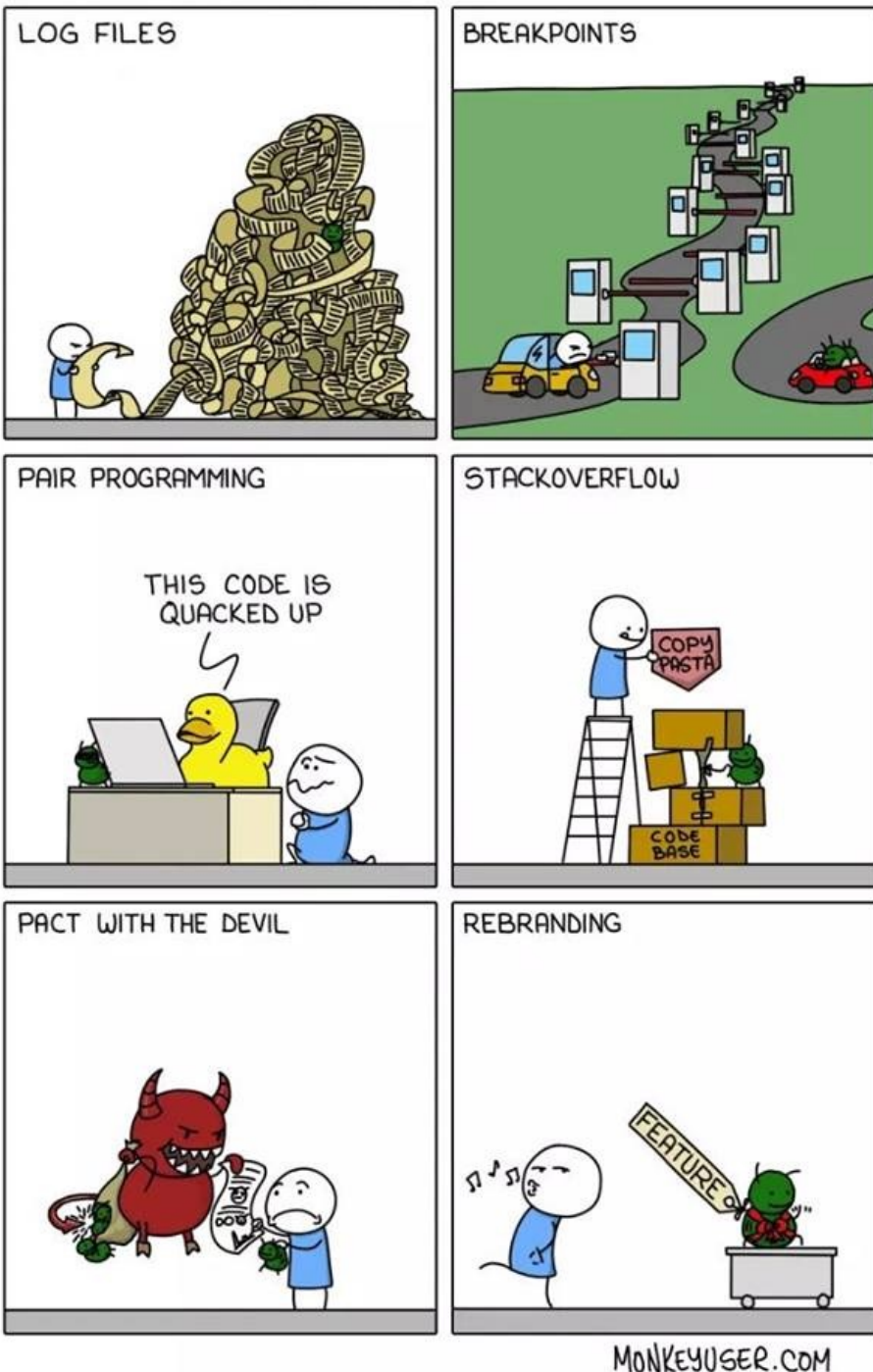
“

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

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

E. Dijkstra

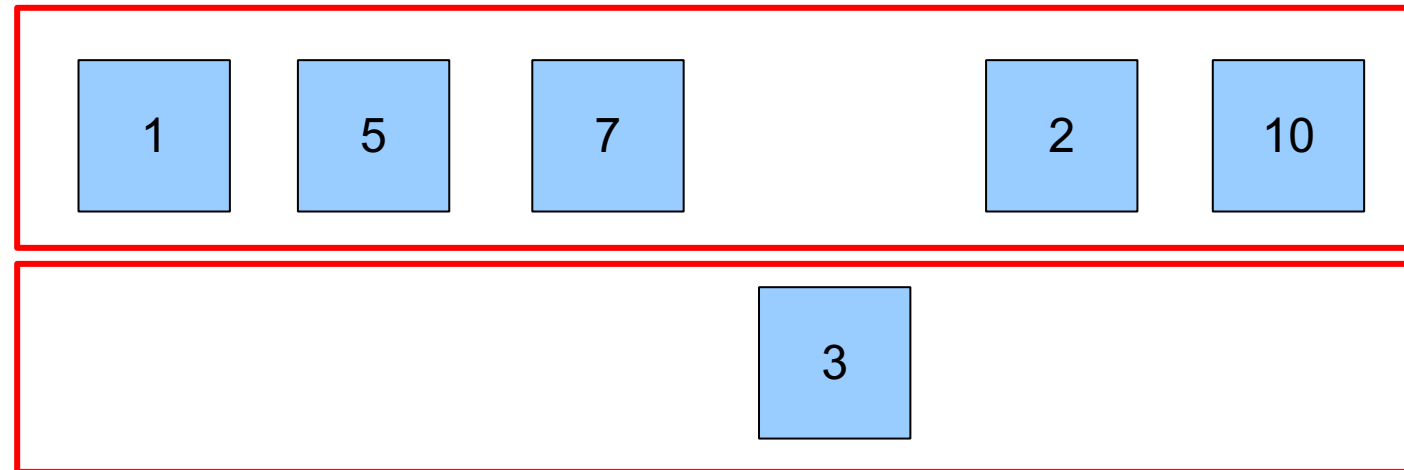
BUG FIXING WAYS



Tecniche

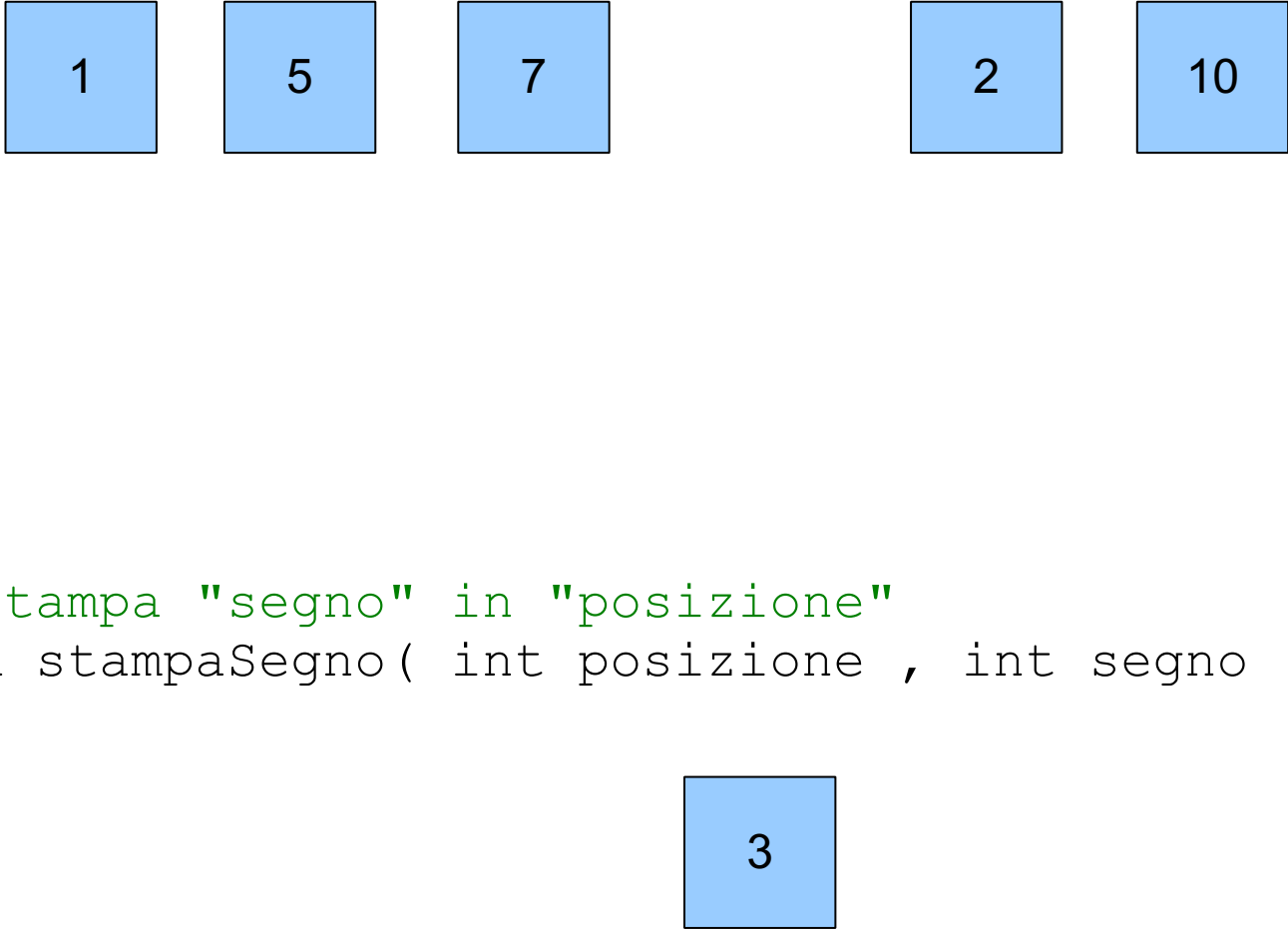
- Testo 
- Visuale
- “Debugger” (es **GDB** , **DDD**)
- Compilatore
- Analisi Memoria (**Valgrind**)

Debug Visuale



Debug Visuale

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



The diagram illustrates a memory layout for a C program. It shows two functions: `stampaArray` and `stampaSegno`. The `stampaArray` function is shown with an array of five elements: 1, 5, 7, 2, and 10. The `stampaSegno` function is shown with a single element: 3. The numbers are displayed in blue boxes, representing memory cells. The array elements are arranged in a row, with a gap between 7 and 2, suggesting a memory gap or a specific memory layout. The element 3 is shown in a separate box below the array.

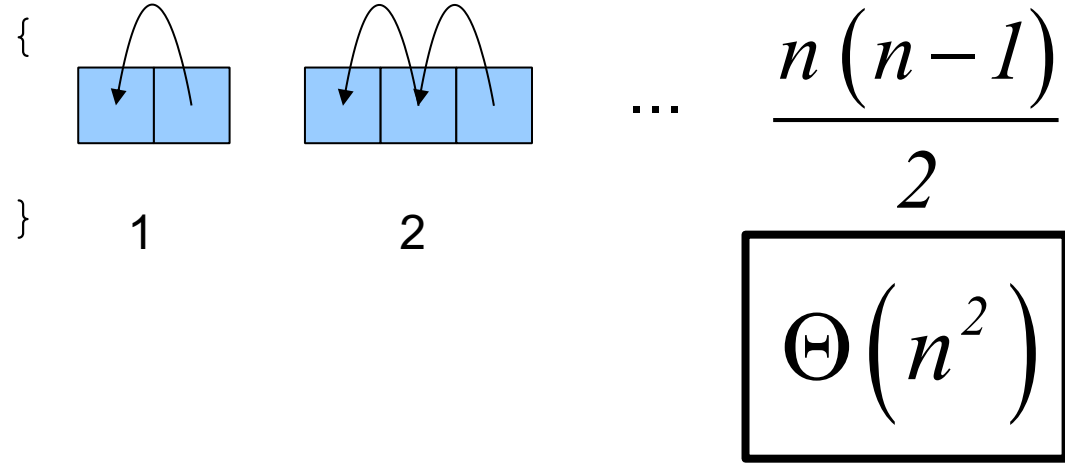
Debug Visuale

```
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 }
```


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             ...  $\frac{n(n-1)}{2}$ 
13
14        }
15
16
17    }
18 }
```

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        total += counter;
16    }
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

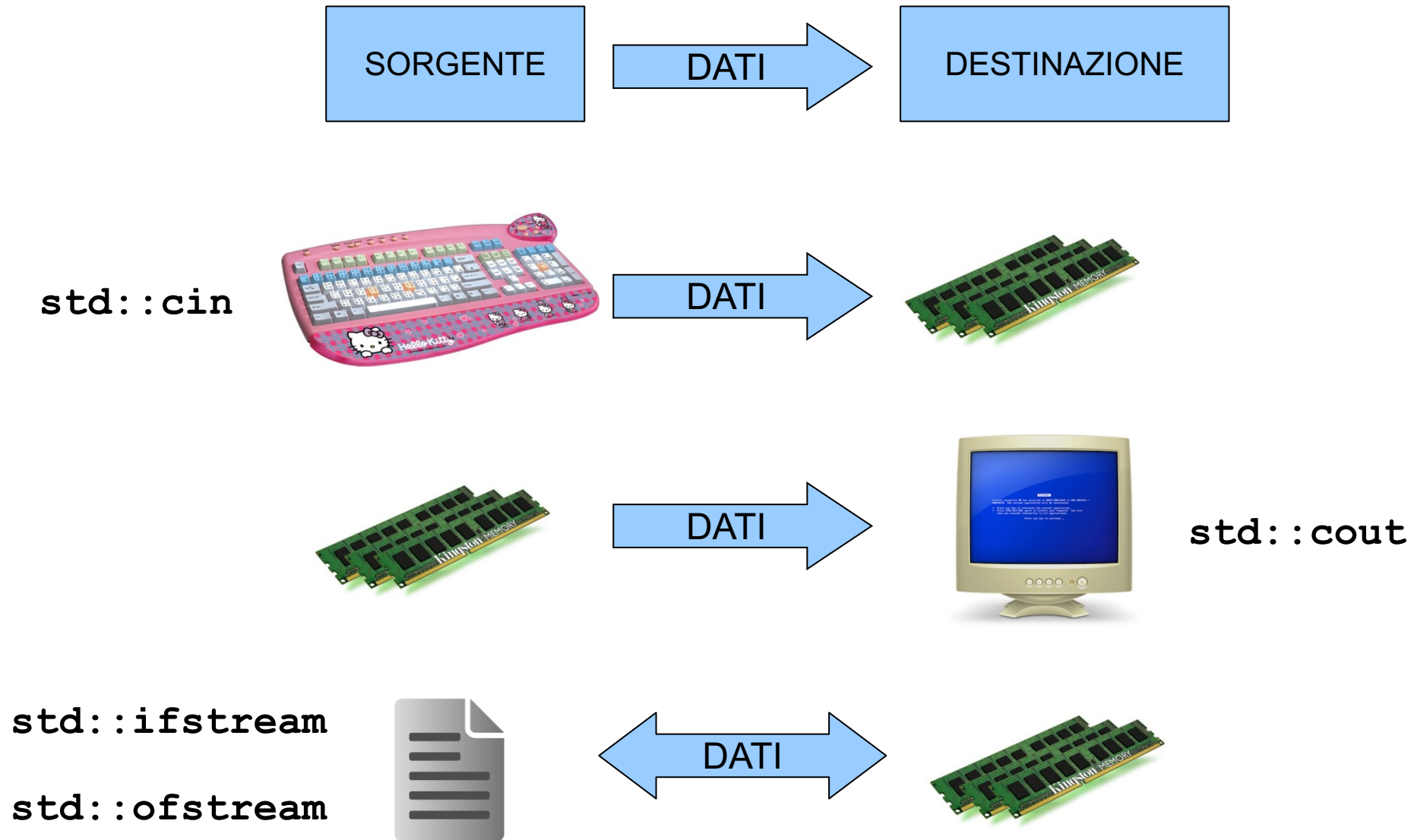


Memoria dinamica

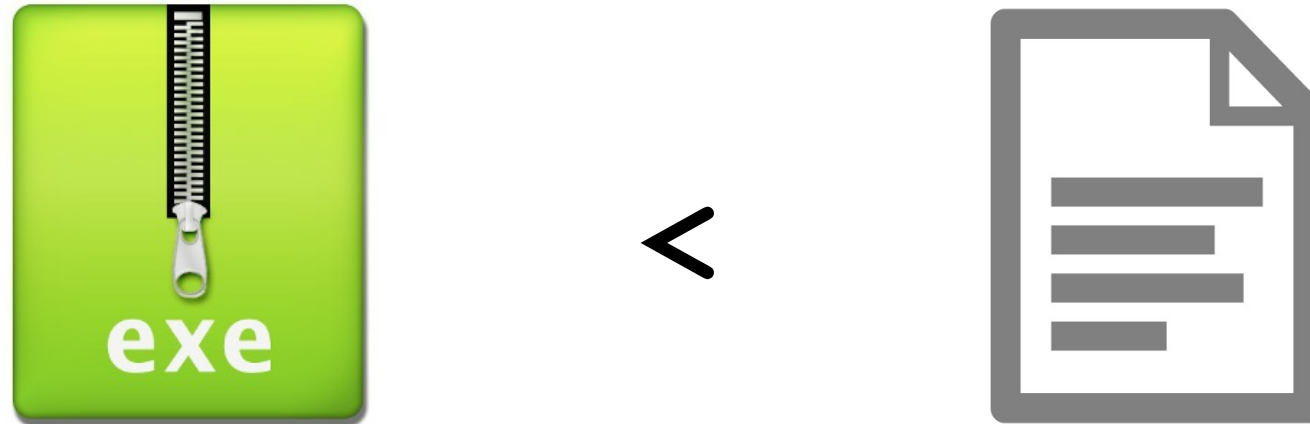


- Quantità di dati **NON** nota a tempo di compilazione
- Quantità di dati **VARIABLE** durante l'esecuzione

Lettura Input



Redirezione DA File



```
./mySort < reqFile
```

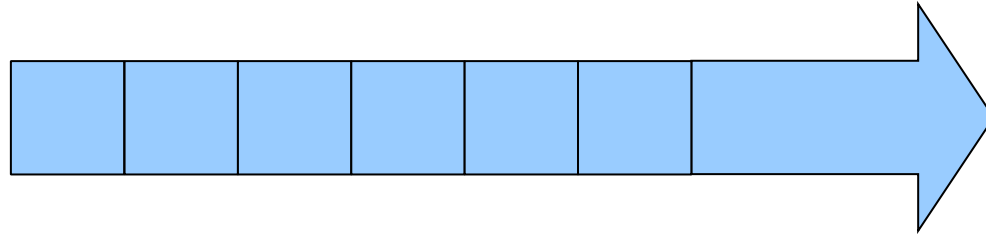
Lettura Input

```
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
```

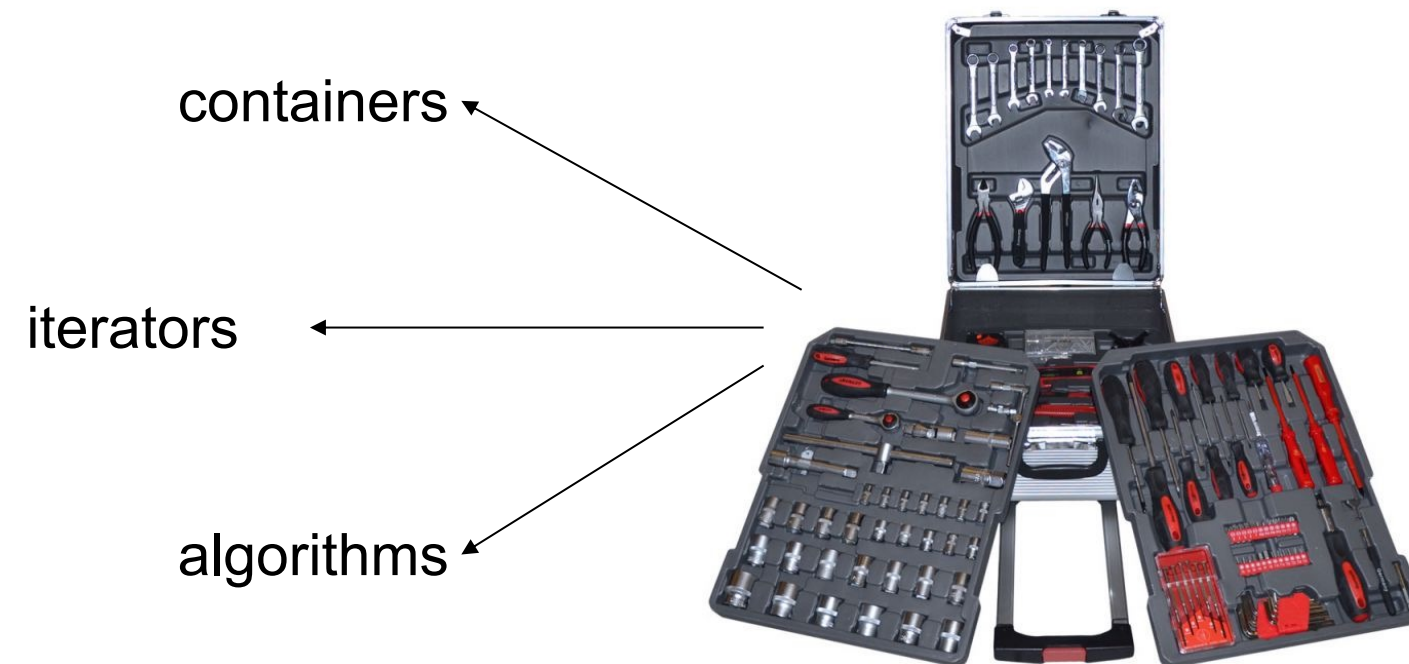
Vettore



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

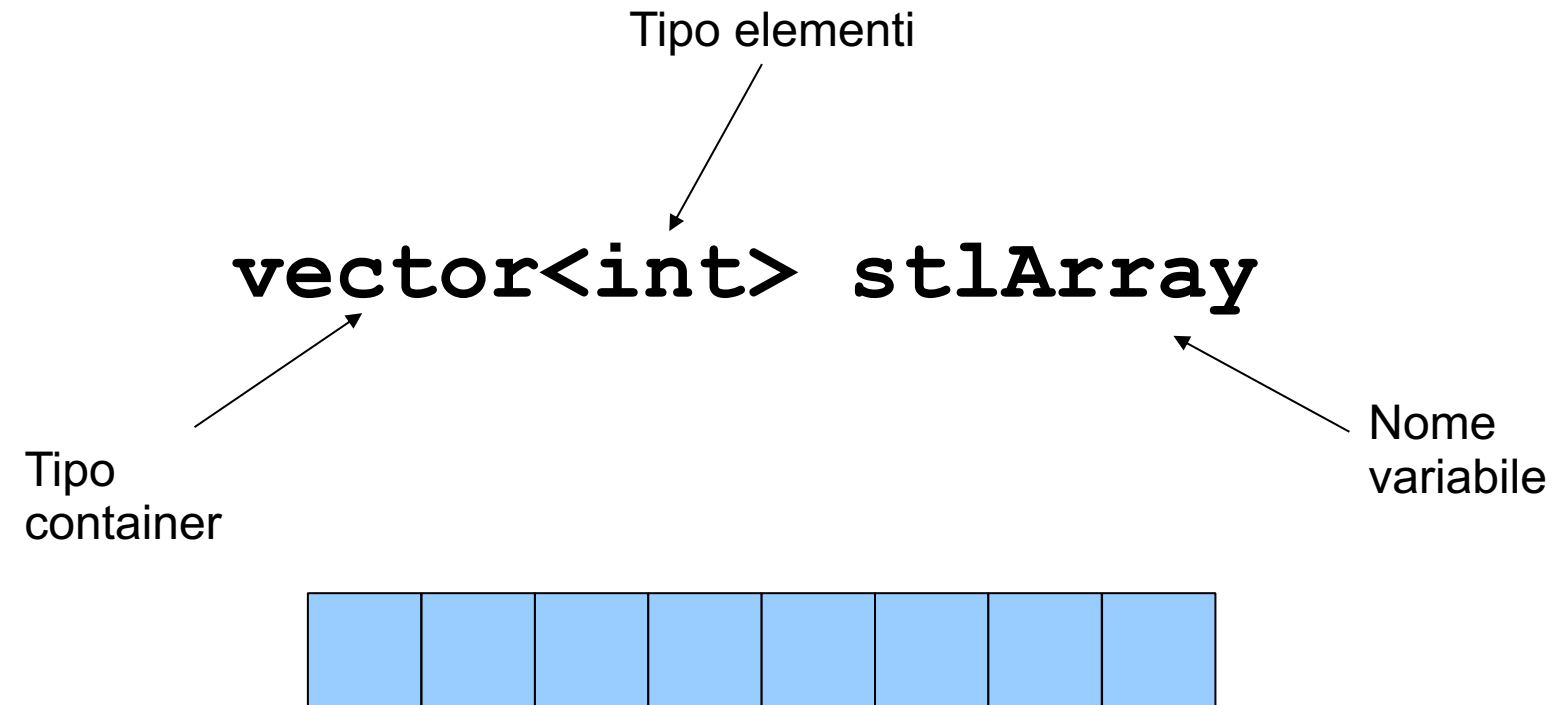


Standard Template Library (STL)



Antonio Virdis - 2022

Uso Vector



Uso Vector

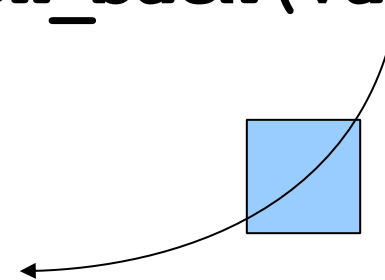
```
vector<int> stlArray
```



Uso Vector

```
vector<int> stlArray
```

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

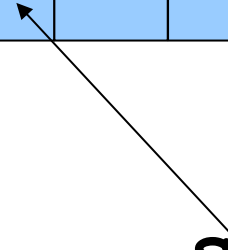


Use Vector

```
vector<int> stlArray
```



`stlArray[i]`

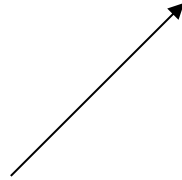


Use Vector

```
vector<int> stlArray
```



`&stlArray[0]`



Uso Vector

```
vector<int> stlArray
```

`stlArray.begin()`

`stlArray.end()`



Use Vector

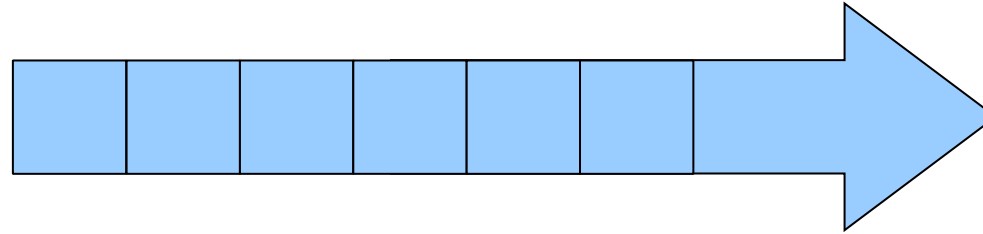
```
vector<int> stlArray
```



`stlArray.size()`

Uso Vector

- Dinamico
 - Allocazione dinamica dimensione



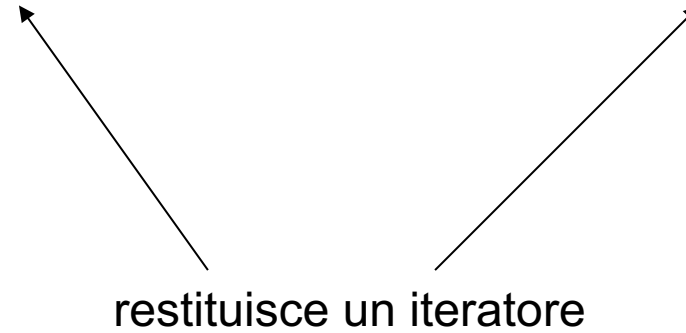
- Contiguo
 - Accesso Random con costo costante
 - Gestione array-like ← **(con prudenza)**

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
```

GENERAZIONE OUTPUT

```
cout << uscita;
```

VERIFICA

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

Esercizio

Input: `input.txt`

```
3
1
9
15
```

Input

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

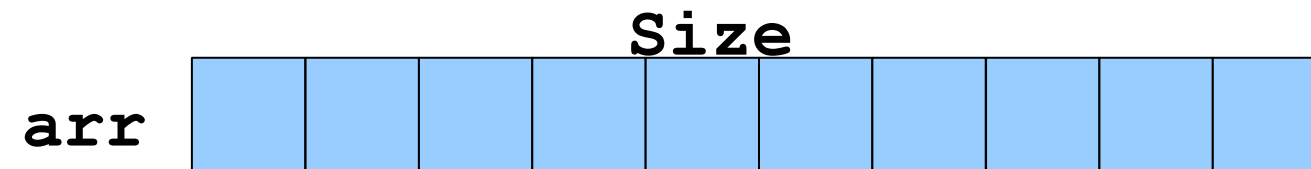
Output: `output.txt`

```
25
135
yes
```

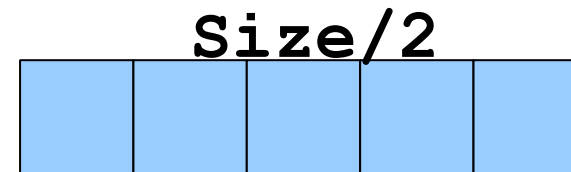
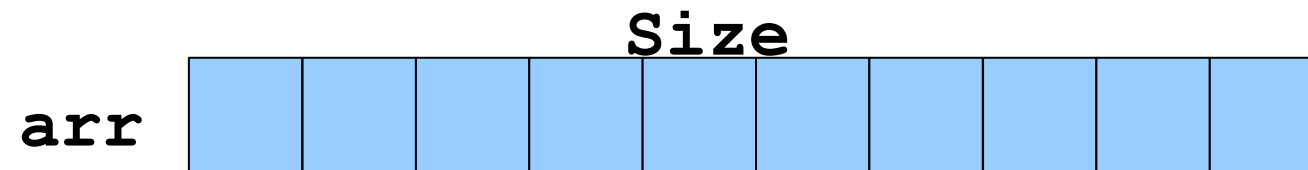
Output

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

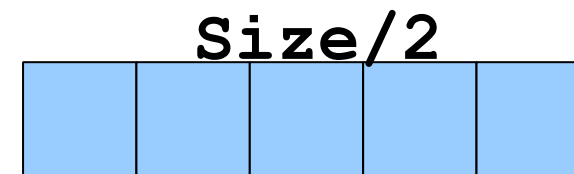
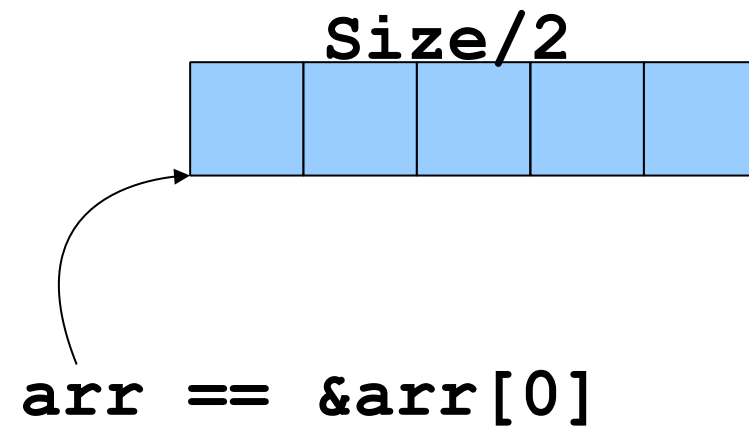
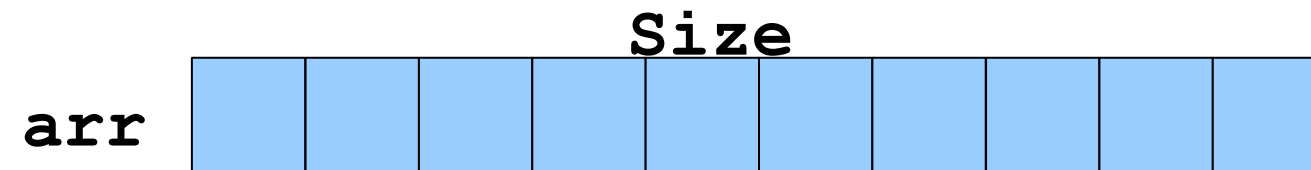
A metà



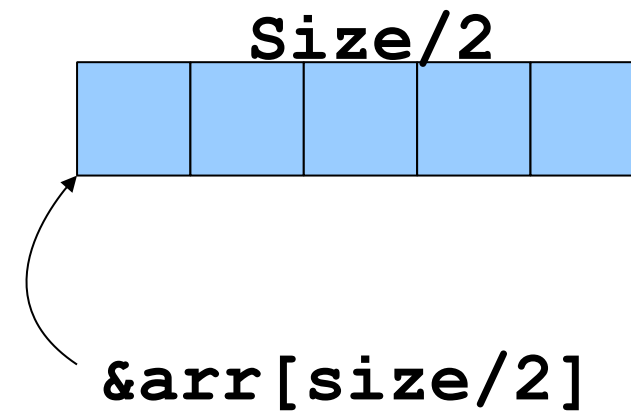
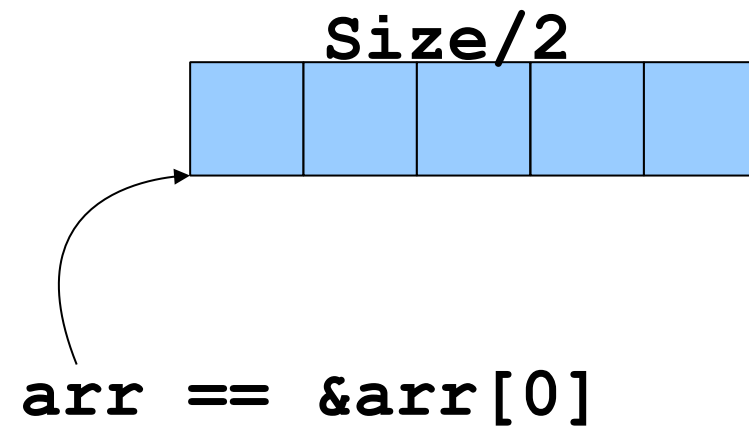
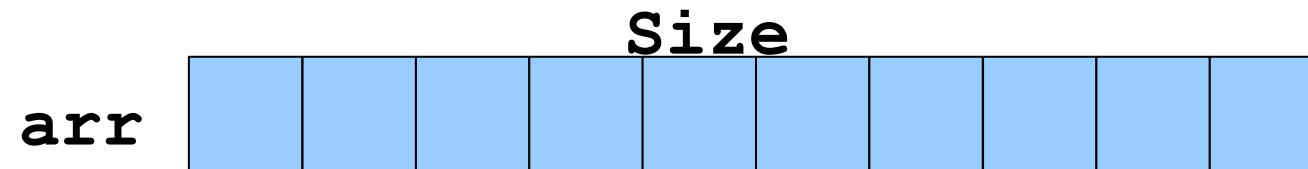
A metà



A metà



A metà



Unire gli array

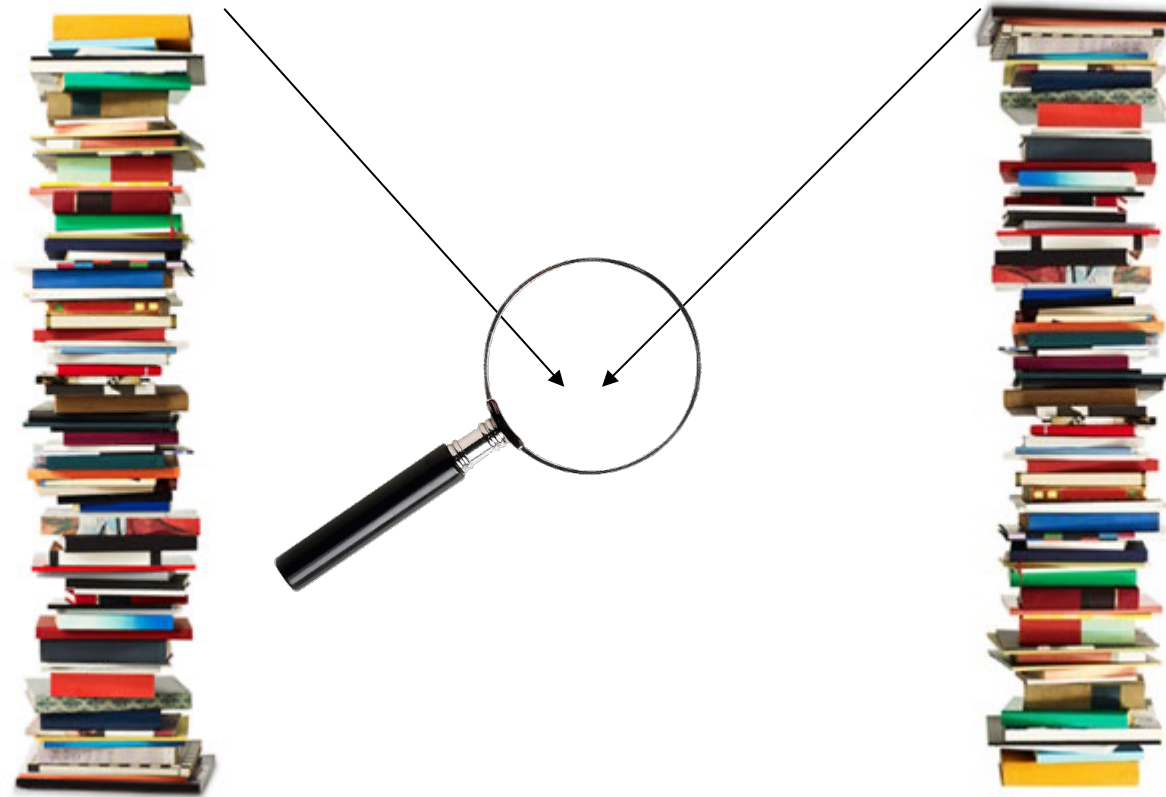


Antonio Virdis - 2022

Unire gli array

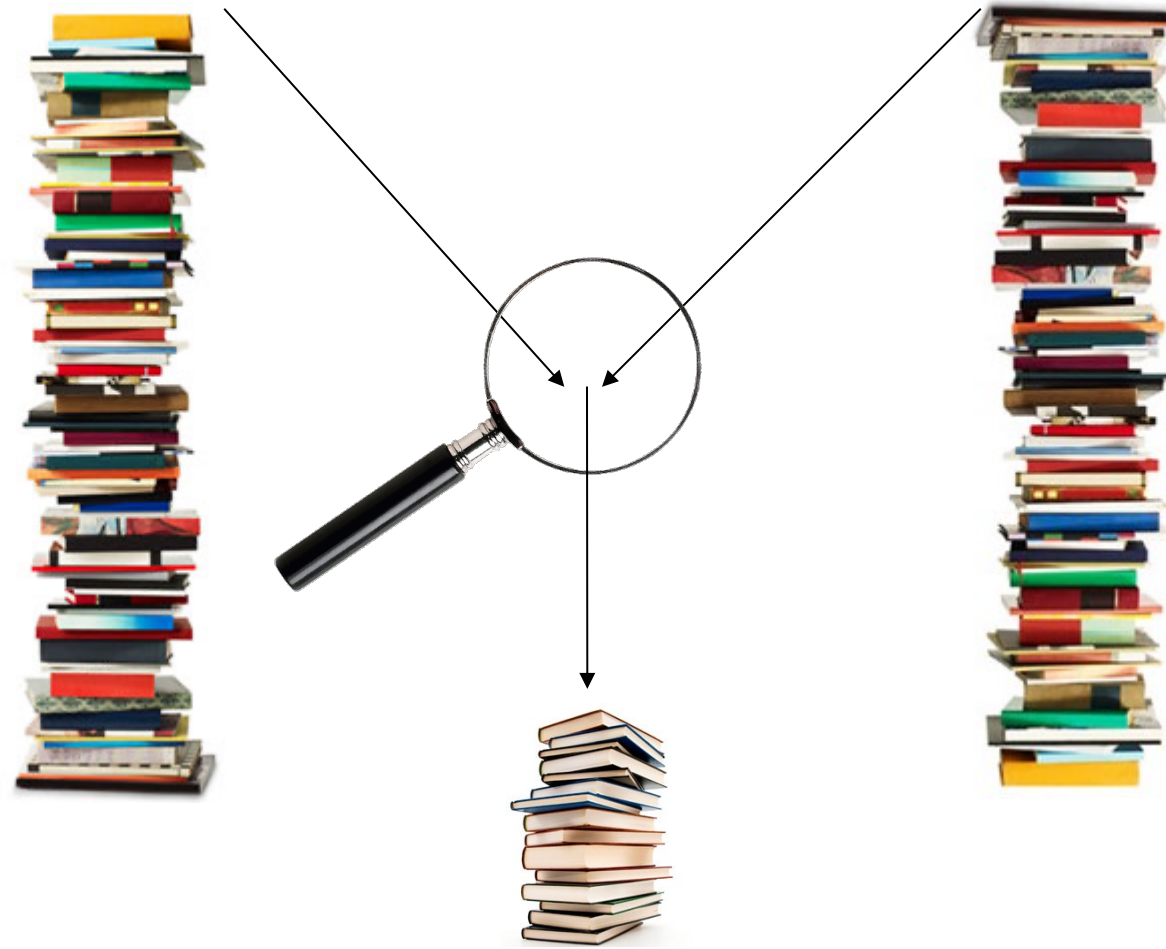


Unire gli array



Antonio Virdis - 2022

Unire gli array



Antonio Virdis - 2022

1	4	9	12	27
---	---	---	----	----

2	3	5	6	8
---	---	---	---	---

1
4
9
12
27

2
3
5
6
8