Algoritmi e Strutture Dati

Lezione 1

http://mlpi.ing.unipi.it/alfeo

Antonio Luca alfeo

luca.alfeo@ing.unipi.it

“Ma io so già programmare!”

Fondamenti I	Fondamenti II
Sia dato un array contenente delle **frasi**. Scrivere un programma che prenda in input una **stringa** e restituisca tutte le frasi che la contengono.	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
- Esame in laboratorio

Pre Requisiti

- Fondamenti I
- Utilizzo compilatore
- Comandi base unix

Sommario

- Debug Triviale
- Gestione Dinamica Input e Liste
- Debug Assistito
- Soluzioni della Standard Template Library
- Gestione Stringhe e Vector

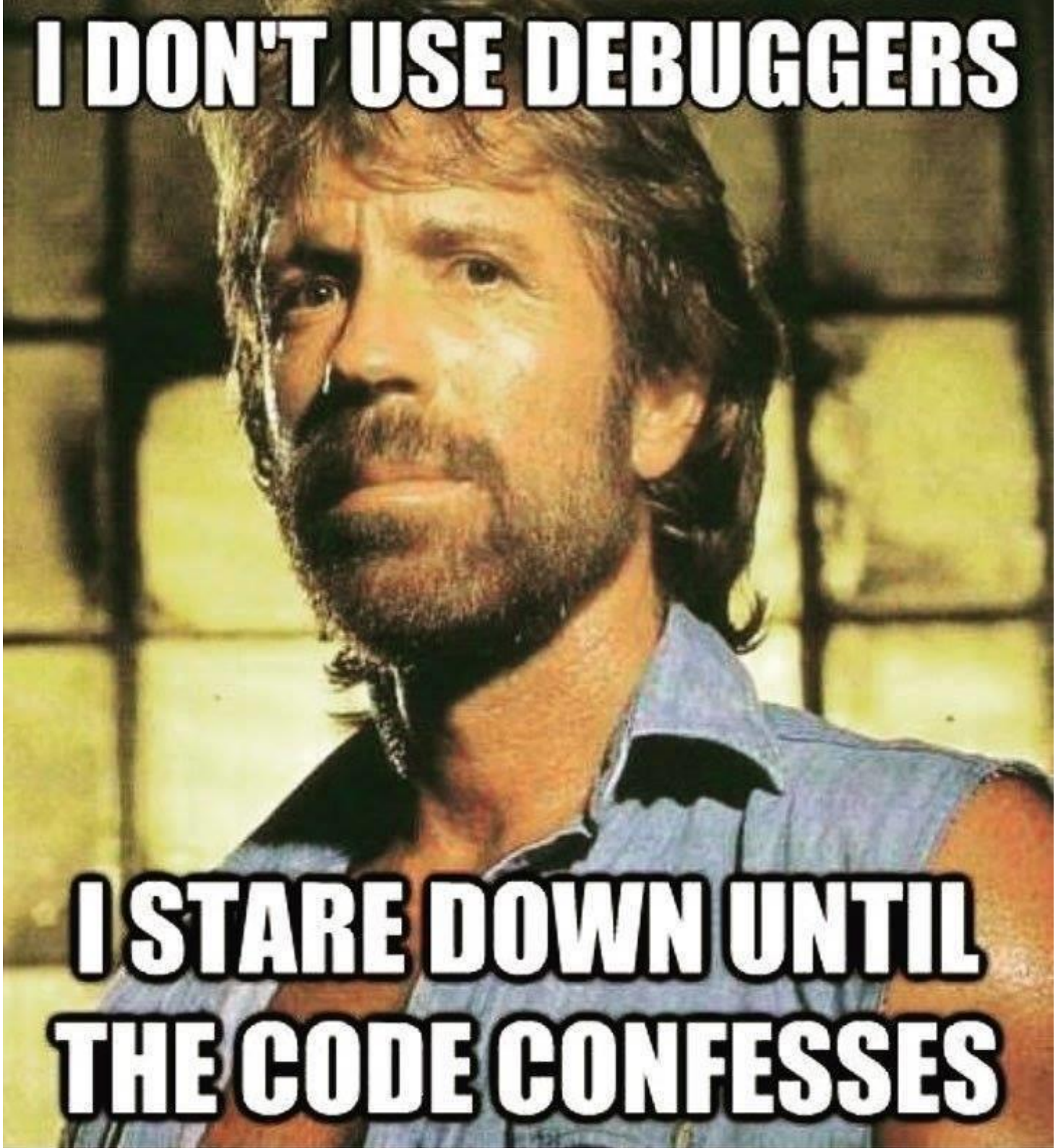
Debugging

“

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

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

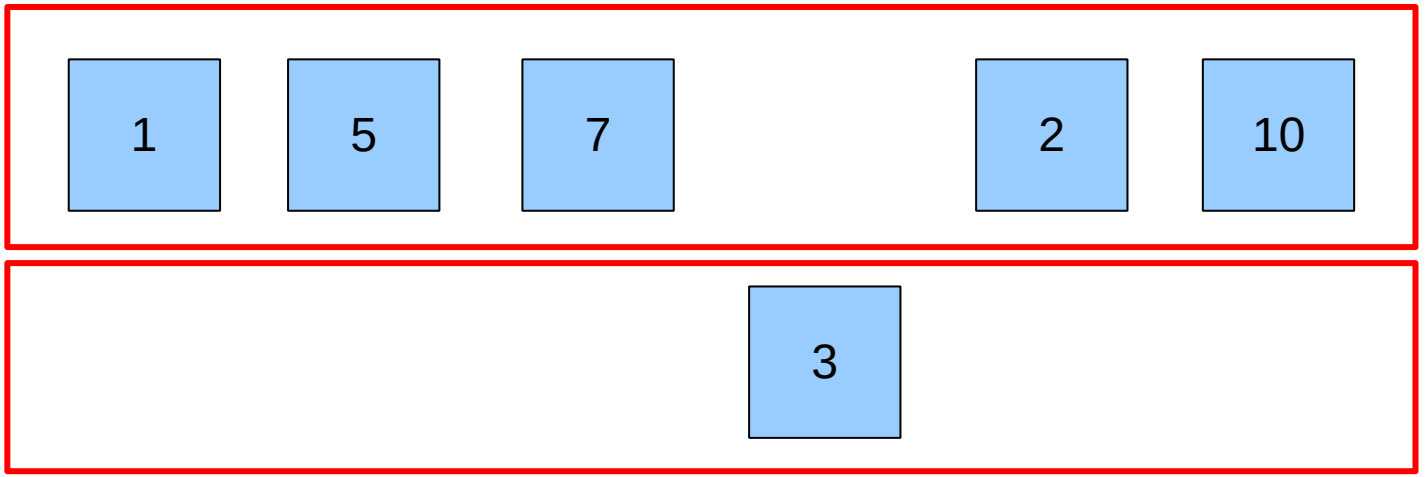
E. Dijkstra


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

Tecniche

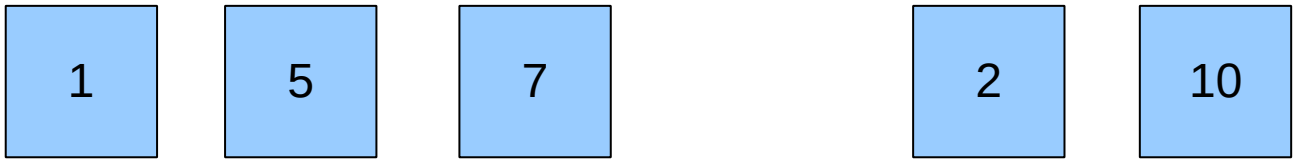
- Testo 
- Visuale
- “Debugger” (es **GDB**)
- Compilatore
- Analisi Memoria (**Valgrind**)

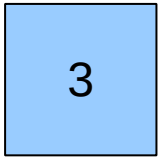
Debug Visuale



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

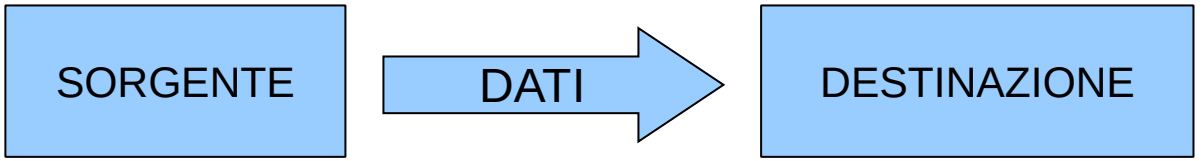
```
1   // stampa con buco
2   void stampaArray( int arr[] , int len , int buco)
3   {
4       for( int i=0 ; i < len ; ++i )
5       {
6           if(i==buco)
7               cout << "\t";
8           else
9               cout << arr[i] << "\t" ;
10      }
11      cout << endl;
12  }
13
14  // stampa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


IN SUPREMÆ DIGNITATIS
1343

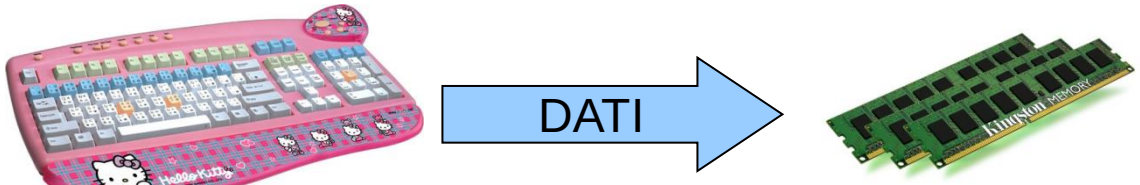
Tipo Accessi

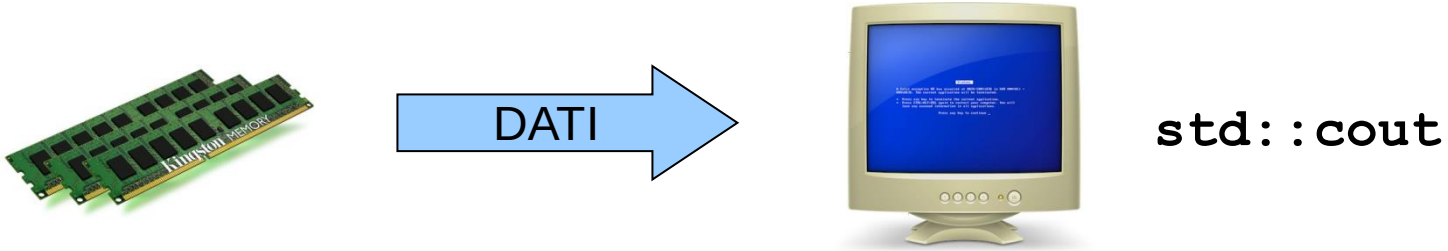
vs

Lettura Input



std::cin

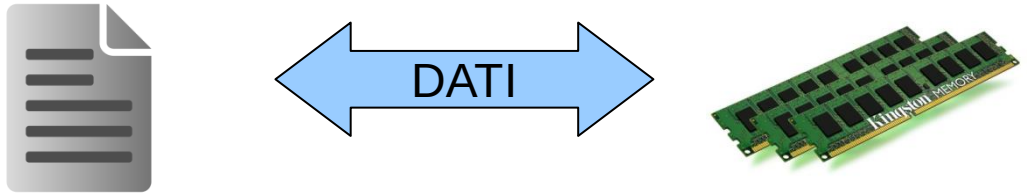




std::cout

std::ifstream

std::ofstream



Redirezione DA File

<

```
./stlSort < reqFile
```

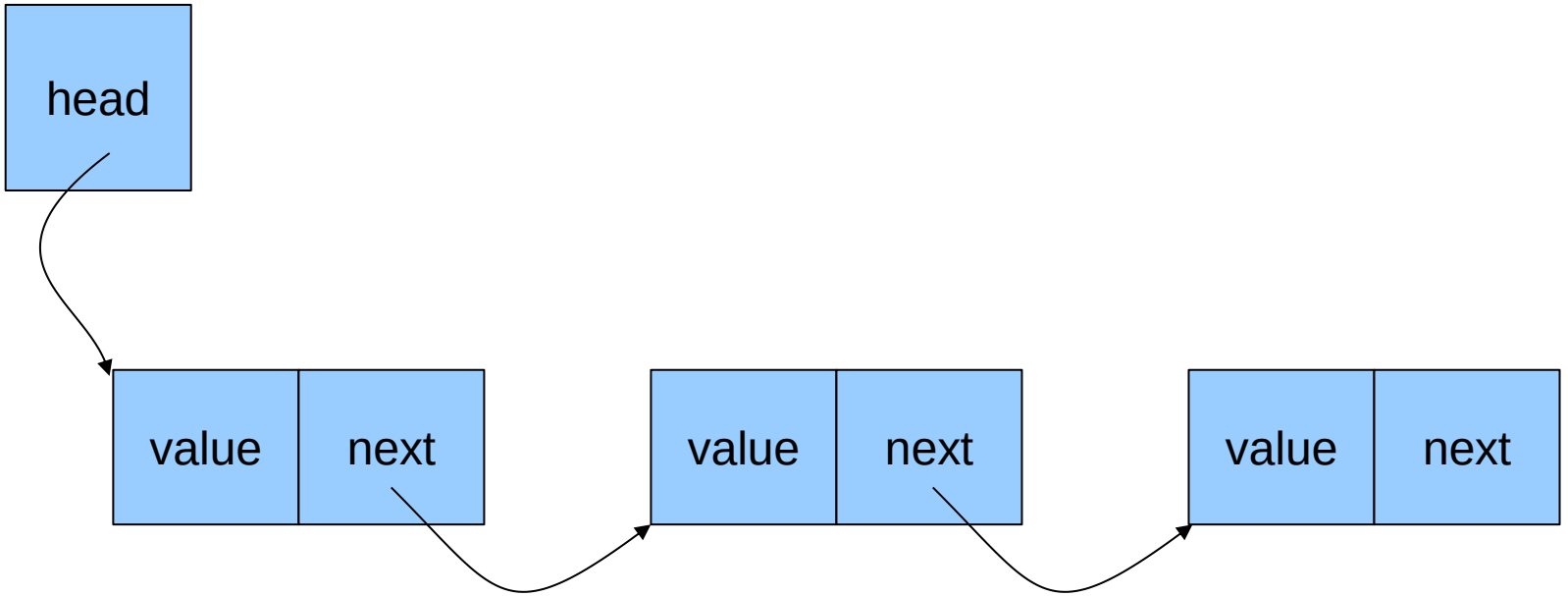
Lettura Input

```
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
```

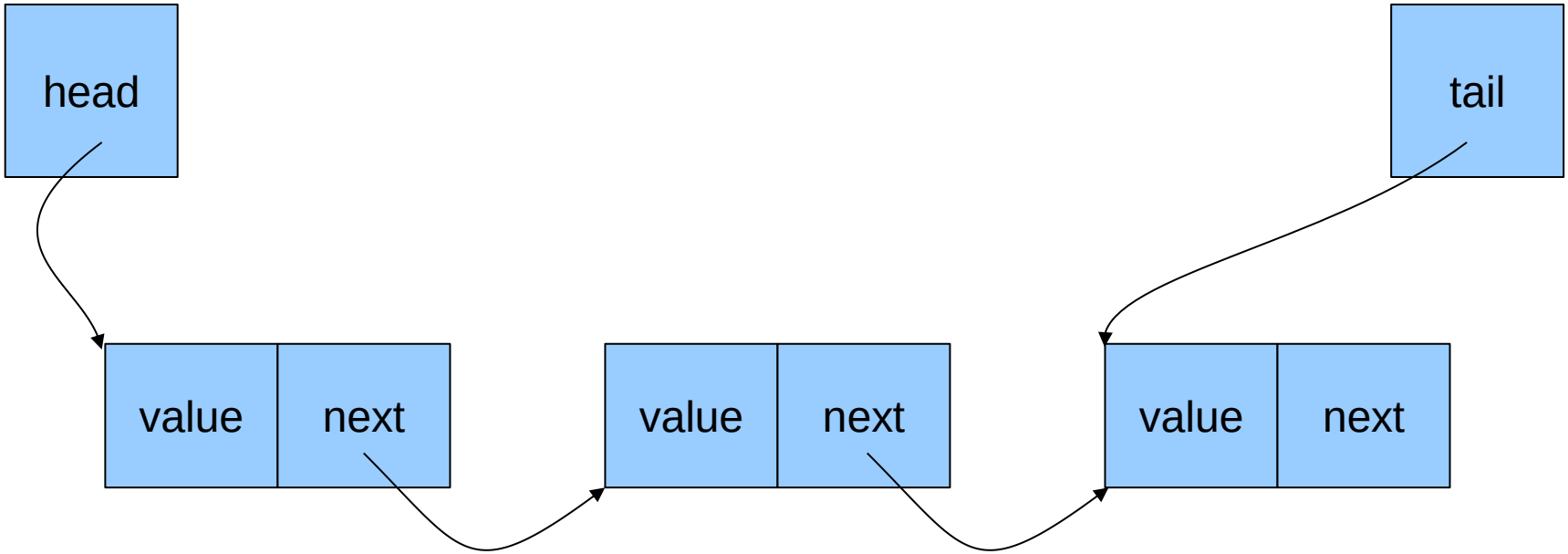
Lettura Input

```
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
```

LISTE



liste



Solo inserimento in coda

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       // LEGGO LUNGHEZZA
4
5
6       // VARIABILI DI APPOGGIO
7
8       // PER TUTTA LA LUNGHEZZA
9       {
10          // LEGGO VALORE
11
12          // CREO E INIZIALIZZO OGGETTO
13
14
15          // AGGIORNO TESTA
16      }
17      // RITORNO TESTA
18  }
```

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , * newObj;
7
8       for( int i = 0 ; i < l ; ++i )
9       {
10          cin >> value;
11          newObj = new Obj();
12          newObj->next_ = head;
13          newObj->value_ = value;
14
15          head = newObj;
16      }
17      return head;
18  }
```

Stampa Lista

```
1   void stampaLista( Obj * head )
2   {
3       Obj * pointer = head;
4       while( pointer != NULL )
5       {
6           cout << pointer->value_ << endl ;
7           pointer = pointer->next_;
8       }
9       cout << endl;
10  }
11
12
```

Stampa Lista

File testList.cpp

```
18  void stampaLista( Obj * head )
19  {
20      Obj * pointer = head;
21      while( pointer != NULL )
22      {
23          cout << pointer->value_ << endl ;
24          pointer = pointer->next_;
25      }
26      cout << endl;
27  }
28
29
```

Lettura su Lista

```
1   Obj * leggiInput()
2   {
3       int value , l;
4       cin >> l;
5
6       Obj * head , * newObj;
7
8       for( int i = 0 ; i < l ; ++i )
9       {
10          cin >> value;
11          newObj = new Obj();
12          newObj->next_ = head;
13          newObj->value_ = value;
14
15          head = newObj;
16      }
17      return head;
18  }
```

Operazioni su Lista

- Ricerco un elemento e lo sposto in testa
  - Scorrere
  - Estrazione
  - Inserimento in testa
- Ricerco un elemento e lo sposto in coda
  - Scorrere
  - Estrazione
  - Inserimento in coda...

Memoria dinamica

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

Birra!

Ballmer's Peak

Fonte: https://xkcd.com/323/

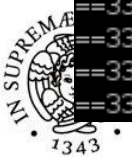
```
kruviser@ilMioComputer:~/Dropbox/lezioni algoritmi/lezione 2$ ./testList < listFile
letto 10
5       1       26      8       12      78      6       2       18      3
3
18
2
6
78
12
8
26
1
5
Segmentation fault (core dumped)
```

Valgrind

- Babysitter Memoria

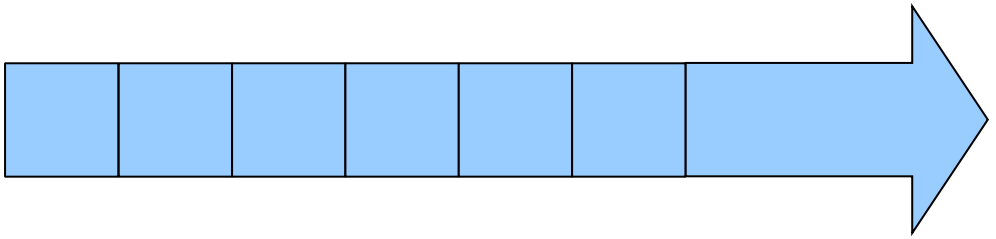
- Controlla accessi
- Conta accessi

```
12
8
26
1
5
==3307== Conditional jump or move depends on uninitialised value(s)
==3307==    at 0x8048719: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==
==3307== Use of uninitialised value of size 4
==3307==    at 0x80486E5: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==
==3307== Invalid read of size 4
==3307==    at 0x80486E5: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==  Address 0xffff is not stack'd, malloc'd or (recently) free'd
==3307==
==3307==
==3307== Process terminating with default action of signal 11 (SIGSEGV)
==3307==  Access not within mapped region at address 0xFFFF
==3307==    at 0x80486E5: stampaLista(Obj*) (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==    by 0x804883D: main (in /home/kruviser/Dropbox/lezioni algoritmi/lezione 2/testList)
==3307==  If you believe this happened as a result of a stack
==3307==  overflow in your program's main thread (unlikely but
==3307==  possible), you can try to increase the size of the
==3307==  main thread stack using the --main-stacksize= flag.
==3307==  The main thread stack size used in this run was 8388608.
==3307==
==3307== HEAP SUMMARY:
==3307==     in use at exit: 80 bytes in 10 blocks
==3307==   total heap usage: 10 allocs, 0 frees, 80 bytes allocated
==3307==
==3307== LEAK SUMMARY:
==3307==    definitely lost: 0 bytes in 0 blocks
==3307==    indirectly lost: 0 bytes in 0 blocks
==3307==      possibly lost: 0 bytes in 0 blocks
==3307==    still reachable: 80 bytes in 10 blocks
```



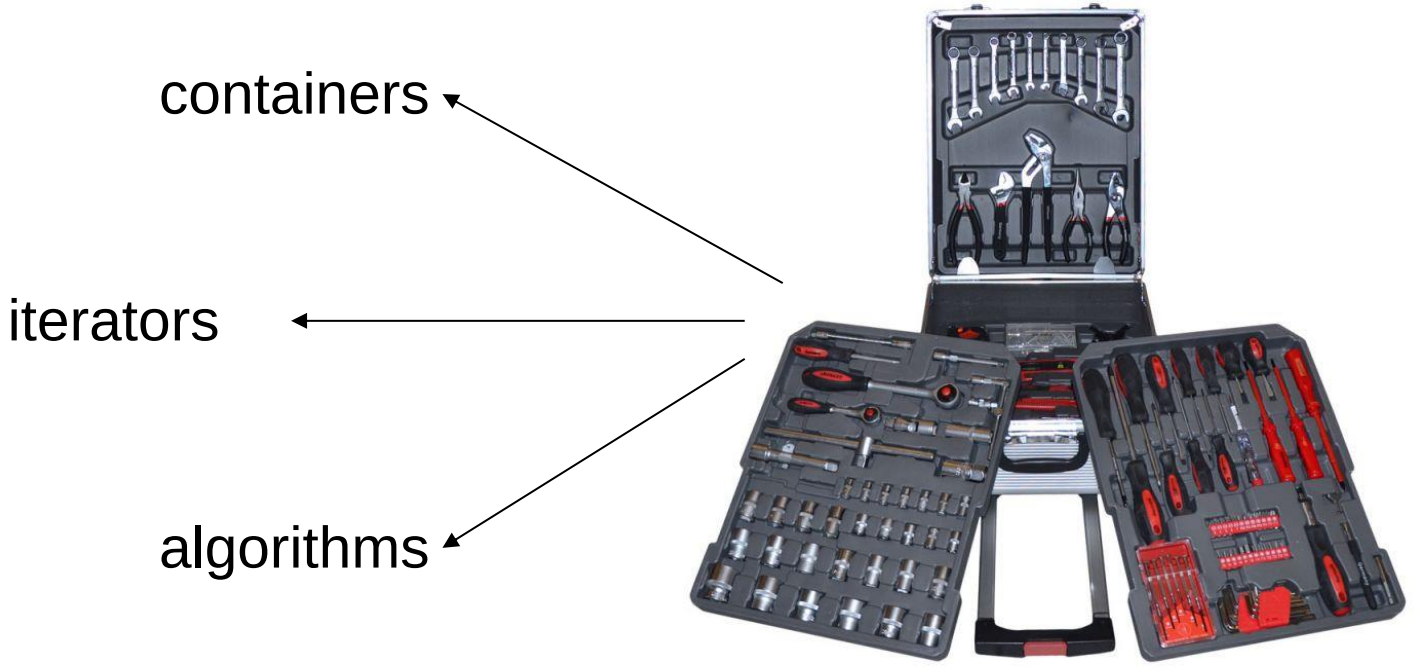
```
12
8
26
1
5
==3288== Conditional jump or move depends on uninitialised value(s)
==3288==    at 0x8048719: stampaLista(Obj*) (testList.cpp:21)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==
==3288== Use of uninitialised value of size 4
==3288==    at 0x80486E5: stampaLista(Obj*) (testList.cpp:23)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==
==3288== Invalid read of size 4
==3288==    at 0x80486E5: stampaLista(Obj*) (testList.cpp:23)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==  Address 0xffff is not stack'd, malloc'd or (recently) free'd
==3288==
==3288==
==3288== Process terminating with default action of signal 11 (SIGSEGV)
==3288==  Access not within mapped region at address 0xFFFF
==3288==    at 0x80486E5: stampaLista(Obj*) (testList.cpp:23)
==3288==    by 0x804883D: main (testList.cpp:58)
==3288==  If you believe this happened as a result of a stack
==3288==  overflow in your program's main thread (unlikely but
==3288==  possible), you can try to increase the size of the
==3288==  main thread stack using the --main-stacksize= flag.
==3288==  The main thread stack size used in this run was 8388608.
==3288==
==3288== HEAP SUMMARY:
==3288==     in use at exit: 80 bytes in 10 blocks
==3288==   total heap usage: 10 allocs, 0 frees, 80 bytes allocated
==3288==
==3288== LEAK SUMMARY:
==3288==    definitely lost: 0 bytes in 0 blocks
==3288==    indirectly lost: 0 bytes in 0 blocks
==3288==      possibly lost: 0 bytes in 0 blocks
==3288==    still reachable: 80 bytes in 10 blocks
```

Vettore



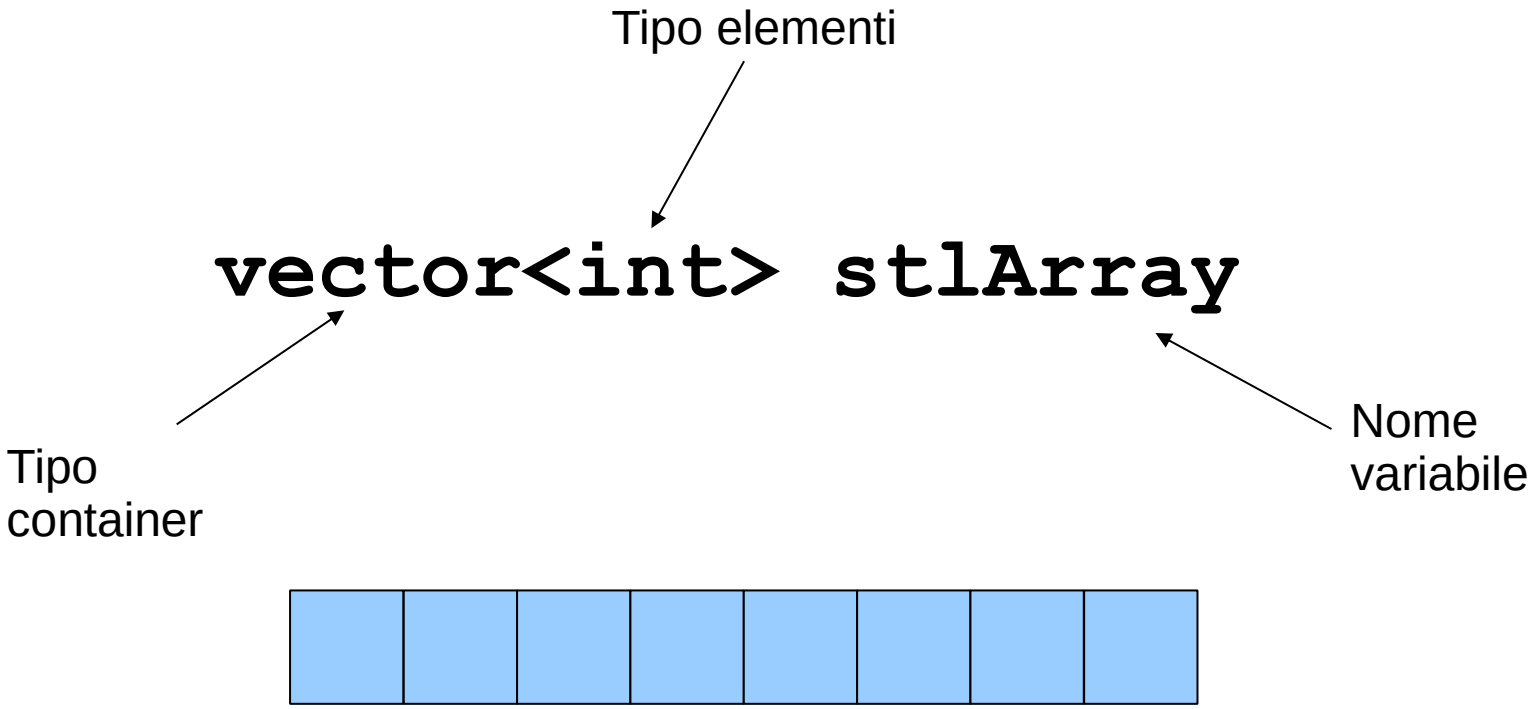
- Struttura dati di dimensione estendibile
- Accesso efficiente
- Algoritmi
- Magari già pronta?!?

Standard Template Library (STL)



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

Uso Vector


Tipo elementi
vector<int> stlArray
Tipo
container
Nome
variabile

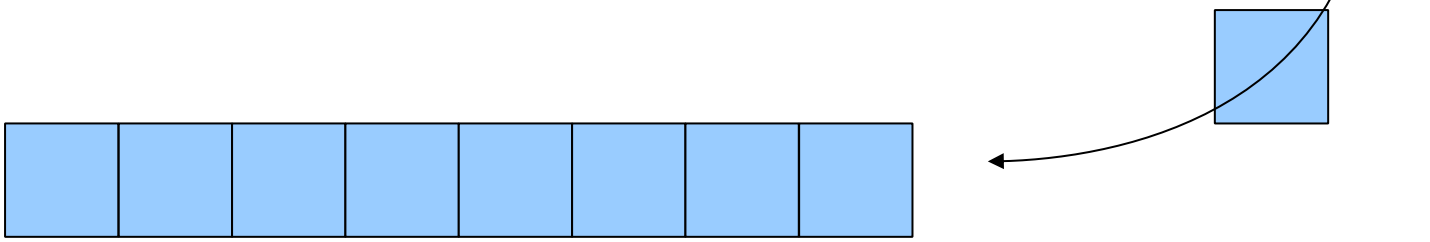
Uso Vector

```
vector<int> stlArray
```

Uso Vector

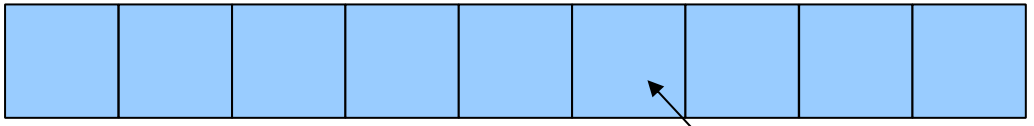
```
vector<int> stlArray
```

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



Uso Vector

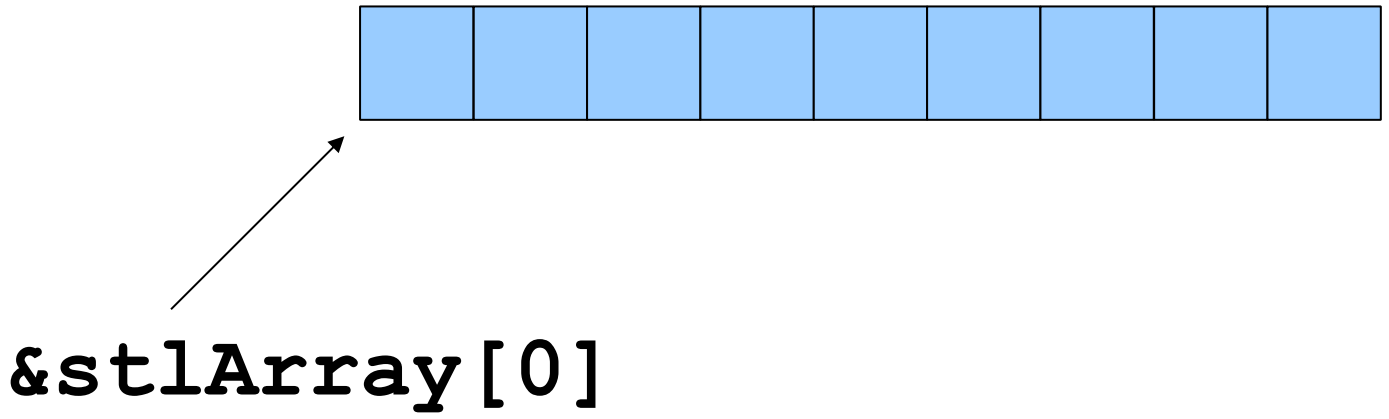
```
vector<int> stlArray
```



```
stlArray[i]
```

Uso Vector

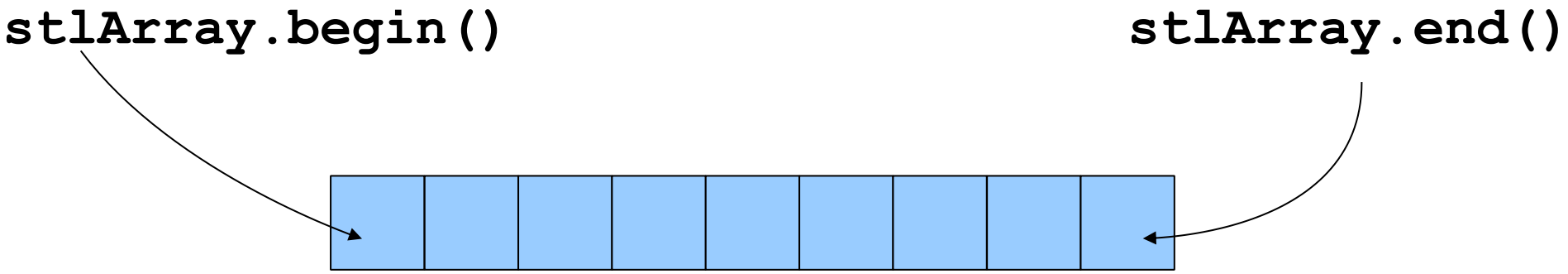
```
vector<int> stlArray
```



```
&stlArray[0]
```

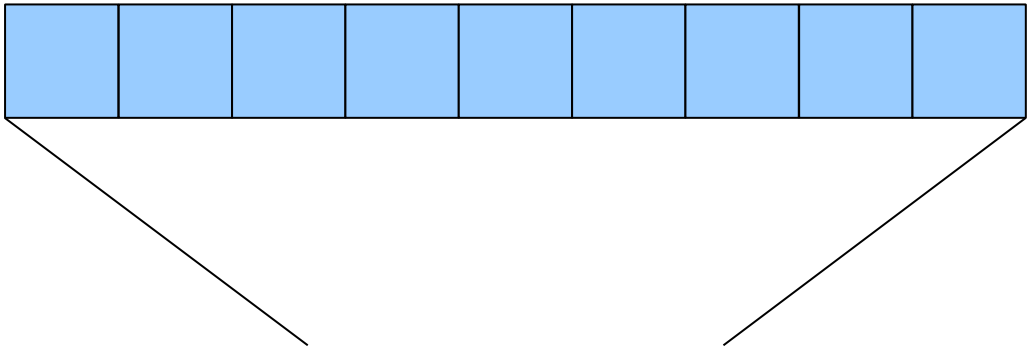
Uso Vector

```
vector<int> stlArray
```



Uso Vector

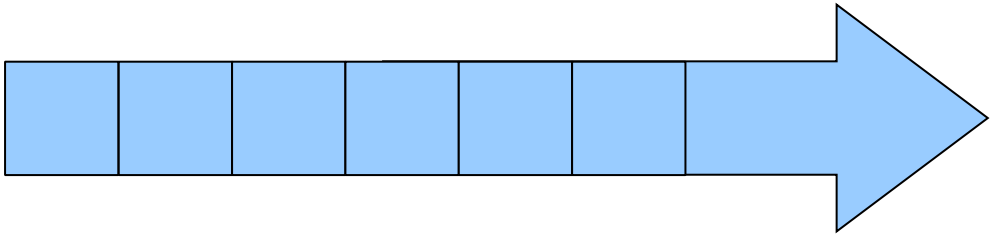
```
vector<int> stlArray
```



```
stlArray.size()
```

Uso Vector

- Dinamico
  - Allocazione dinamica dimensione



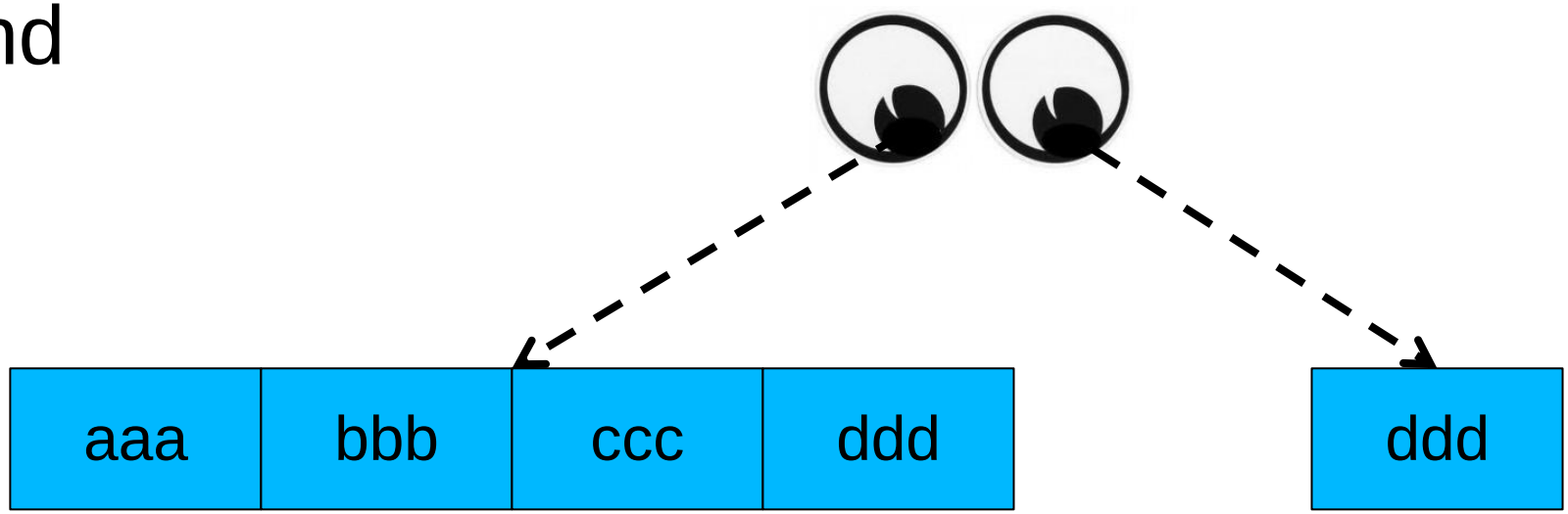
- Contiguo
  - Accesso Random con costo costante
  - Gestione array-like ← **(con prudenza)**

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
```

Stringhe

- Creazione
- Concatenazione
- Compare
- Find



Stringhe

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
#include <string>

String parola = "liste";
```

Stringhe

```
1   #include <string>
2
3   String parola = "liste";
4
5   String frase = "mi piacciono le liste";
6
7
8
9
10
11
12
13
14
```

Stringhe

```
1   #include <string>
2
3   String parola = "liste";
4
5   String frase = "mi piacciono le liste";
6
7   String parola2 = "non ";
8
9   String frase2 = parola2 + frase;
10
11
12
13
14
```

Stringhe

```
1   #include <string>
2
3   String parola = "liste";
4
5   String frase = "mi piacciono le liste";
6
7   String parola2 = "non ";
8
9   String frase2 = parola2 + frase;
10
11  frase.find(parola);
12  // se fallisce -> string::npos
13
14
```

Stringhe

```
1   #include <string>
2
3   String parola = "liste";
4
5   String frase = "mi piacciono le liste";
6
7   String parola2 = "non ";
8
9   String frase2 = parola2 + frase;
10
11  frase.find(parola);
12  // se fallisce -> string::npos
13
14  parola.compare(parola2);
```

http://www.cplusplus.com/reference/string/string/

Esercizio Stringhe

- Input
  - Una testo T formato da più parole
  - Un insieme S di N parole

- Output:
  1. Le parole di S *contenute* in T, ordinate per posizione in T (insieme R1)
  2. Le parole di S *non contenute* in T, in ordine lessicografico (insieme R2)

Analisi

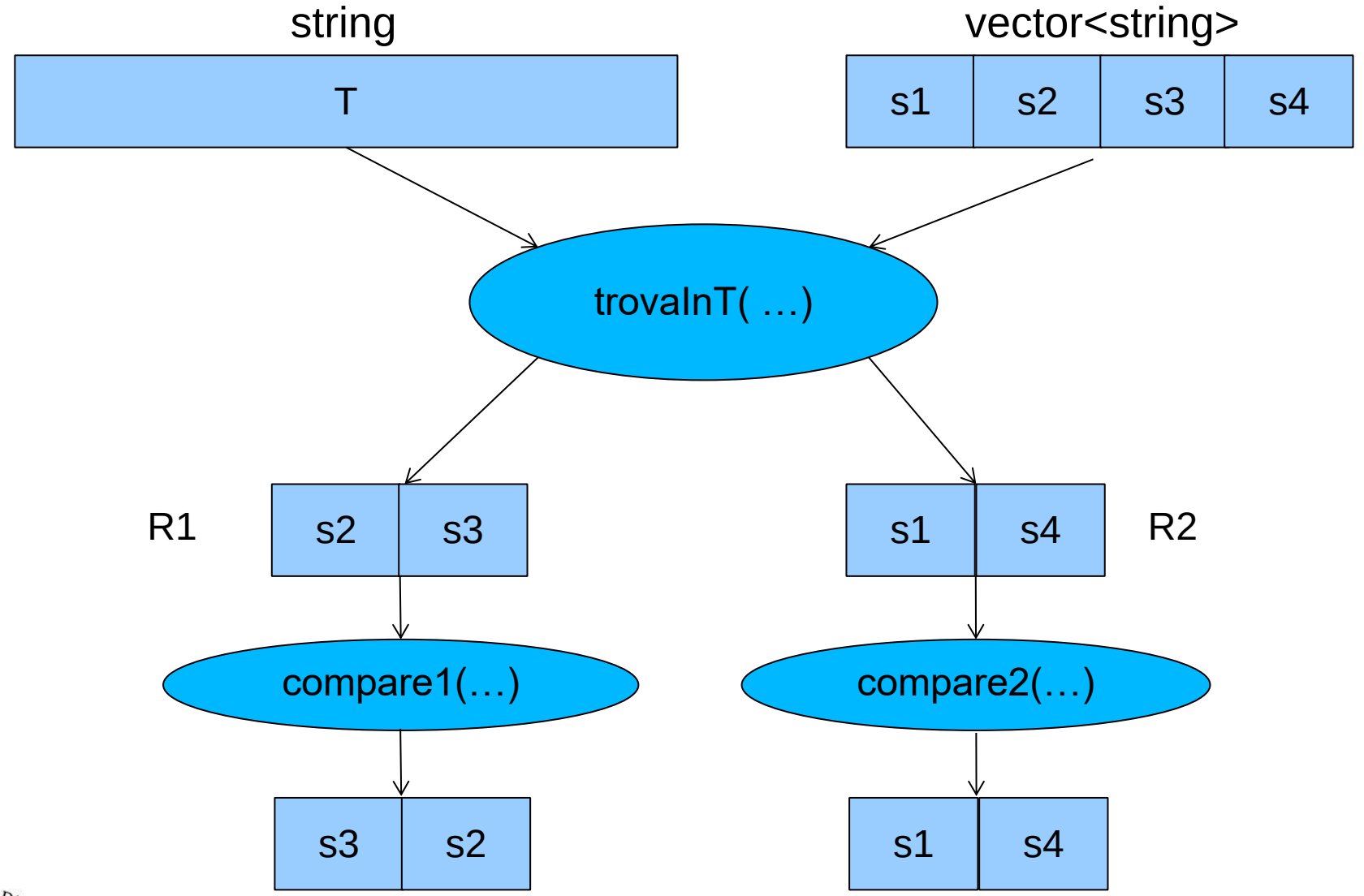
Strutture Dati:

- Dove salvo T?
- Dove salvo S?

Operazioni:

- Come ottengo gli elementi di 1?
- Come ottengo gli elementi di 2?
- Come ordino 1?
- Come ordino 2?

Analisi (2)


string
vector<string>
T
s1
s2
s3
s4
trovaInT(…)
R1
s2
s3
s1
s4
R2
compare1(…)
compare2(…)
s3
s2
s1
s4

Implementazione

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Implementazione

```
1   // cerca stringhe di S dentro T
2   void trovaInT( … ){    }
3
4   // implementa confronto per posizione
5   bool compare1(string a, string b){   }
6
7   // implementa confronto lessicografico
8   bool compare2(string a, string b){   }
9
10
11
12
13
14
15
16
17
18
19
20
```

Implementazione

```
1   // cerca stringhe di S dentro T
2   void trovaInT( … ){    }
3
4   // implementa confronto per posizione
5   bool compare1(string a, string b){   }
6
7   // implementa confronto lessicografico
8   bool compare2(string a, string b){   }
9
10  int main()
11  {
12      string T;
13      vector <string> S, R1, R2;
14
15
16
17
18
19
20
    }
```

Implementazione

```
1   // cerca stringhe di S dentro T
2   void trovaInT( … ){    }
3
4   // implementa confronto per posizione
5   bool compare1(string a, string b){   }
6
7   // implementa confronto lessicografico
8   bool compare2(string a, string b){   }
9
10  int main()
11  {
12      string T;
13      vector <string> S, R1, R2;
14
15      // lettura T ed S
16
17      trovaInT( … );
18      sort( R1.begin(), R1.end(), compare1 )
19      sort( R2.begin(), R2.end(), compare2 )
20
    }
```

Implementazione

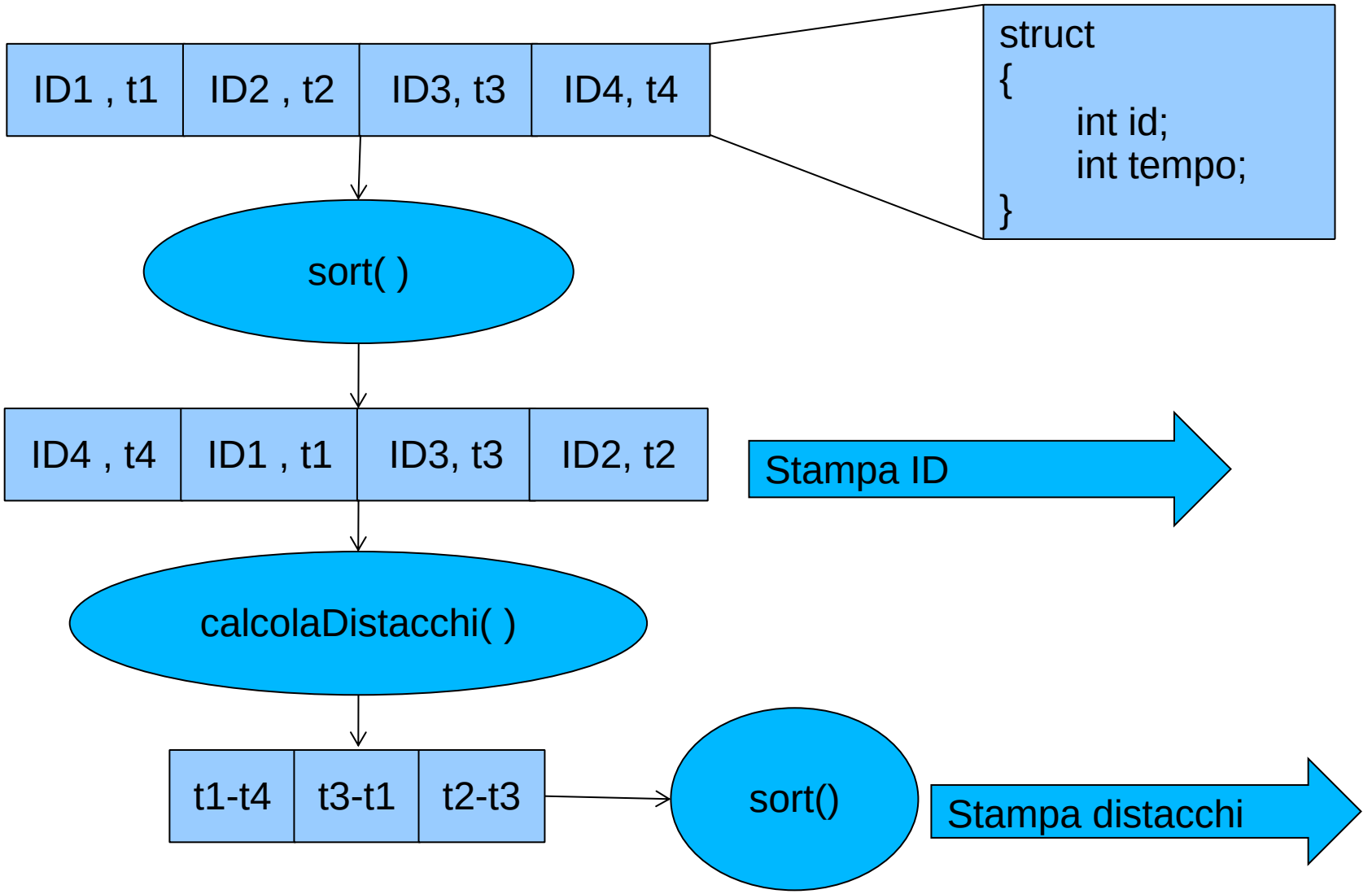
```
1   // cerca stringhe di S dentro T
2   void trovaInT( … ){    }
3
4   // implementa confronto per posizione
5   bool compare1(string a, string b){   }
6
7   // implementa confronto lessicografico
8   bool compare2(string a, string b){   }
9
10  int main()
11  {
12      string T;
13      vector <string> S, R1, R2;
14
15      // lettura T ed S
16
17      trovaInT( … );
18      sort( R1.begin(), R1.end(), compare1 )
19      sort( R2.begin(), R2.end(), compare2 )
20      print();
    }
```

Gara

- Ad una gara partecipano N concorrenti
- Ogni concorrente e' caratterizzato da:
  - Un ID intero
  - Un tempo di arrivo espresso in secondi

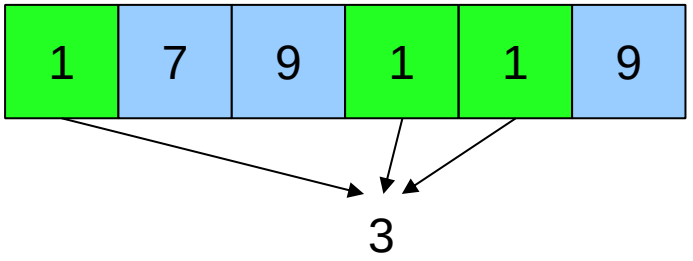
- Calcolare:
  - Classifica
  - K distacchi più ampi di utenti **consecutivi**

Analisi


ID1 , t1
ID2 , t2
ID3, t3
ID4, t4
struct
{
int id;
int tempo;
}
sort( )
ID4 , t4
ID1 , t1
ID3, t3
ID2, t2
Stampa ID
calcolaDistacchi( )
t1-t4
t3-t1
t2-t3
sort()
Stampa distacchi

ESERCIZI:

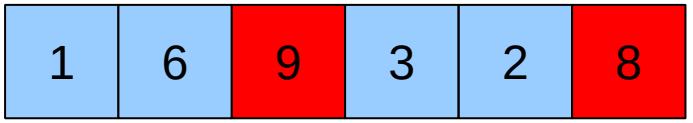
K interi più frequenti



- Input: elementi array , intero k
- Output: primi k valori più frequenti

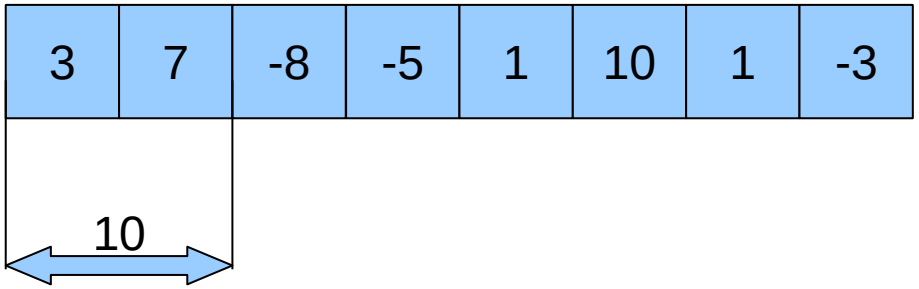
K interi più grandi

K=2



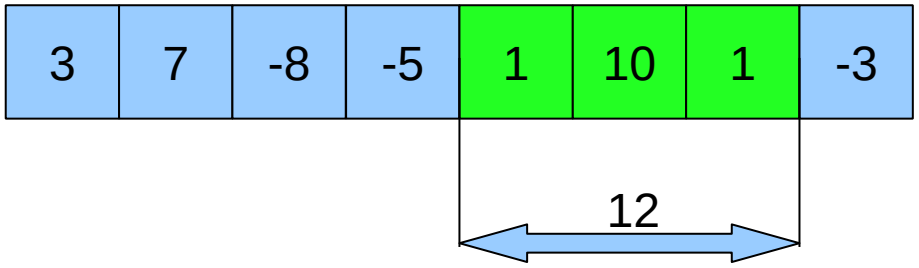
- Input: elementi array , intero k
- Output: primi k valori ordinati in maniera decrescente

Esercizio: Somma Massima



- Input: array
- Output:somma massima di elementi consecutivi

- Esempio



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