Laboratorio di Programmazione

Lezione 4 – Array e stringhe

Marco Anisetti (teoria)
Dipartimento di Informatica
marco.anisetti@unimi.it

Matteo Luperto (lab. turno A)
Dipartimento di Informatica
matteo.luperto@unimi.it

Nicola Bena (lab. turno B)
Dipartimento di Informatica
nicola.bena@unimi.it

Trova l'errore:

```c
#include <stdio.h>

int main() {
int i,j;
int n;
int flag = 0;
do {
printf("Inserisci una cifra massima per calcolare i numeri primi: ");
scanf("%d", &n);
} while (n<=0);

printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2); {
for (j = 2; j * j <= i; j++)
if (n % j == 0)
flag = 0; // Il numero non è primo se è divisibile per i

if (flag) {
printf("%d ", i);
}
}

printf("\n");

return 0;
}
```

Questo programma deve stampare a schermo i numeri primi fino ad `n`, con `n` inserito dall'utente. Purtroppo però ci sono 3 errori. Riuscite a debuggarlo?

Trova l'errore - debug:

```
int main() {
int i,j;
int n = 4; // stampa 2,3;
int flag = 0;

printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2); {
        for (j = 2; j * j <= i; j++)
                if (n % j == 0)
                        flag = 0;

        if (flag) {
                printf("%d ", i);
        }
}

printf("\n");

return 0;
}
```

Step 1: semplifico il flusso del programma (elimino la `scanf` iniziale), indento il codice, trovo un caso di base semplice (es: `n=4`)

Trova l’errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2); {
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }

        if (flag) {
                printf("%d ", i);
        }
}
```

Step 2: verifico il flusso del programma, capisco se entra in tutti i cicli e se le condizioni di entrata sono giuste o meno

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 4: 2
Primo ciclo i=5
Secondo ciclo i,j=5,2
```

Trova l'errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2); {
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }

        if (flag) {
                printf("%d ", i);
        }
}
```

Step 2: verifico il flusso del programma, capisco se entra in tutti i cicli e se le condizioni di entrata sono giuste o meno

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 4: 2
Primo ciclo i=5
Secondo ciclo i,j=5,2
```

Attenzione, entra una sola volta nel primo ciclo con un valore di `i=5`, perché? Controllo il primo ciclo, trovo il primo errore

Trova l'errore - debug:

Primo errore trovato!

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2); {
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }

        if (flag) {
                printf("%d ", i);
        }
}
```

Step 2: verifico il flusso del programma, capisco se entra in tutti i cicli e se le condizioni di entrata sono giuste o meno

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 4: 2
Primo ciclo i=5
Secondo ciclo i,j=5,2
```

Attenzione, entra una sola volta nel primo ciclo con un valore di `i=5`, perché? Controllo il primo ciclo, trovo il primo errore

Trova l’errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2) {
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }

        if (flag) {
                printf("%d ", i);
        }
}
```

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 4: 2
Primo ciclo i=3
```

Con `n=4` entra solo una volta nel primo ciclo, non è sufficiente a capire molto, aumento `n=5`

Trova l’errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2) {
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }

        if (flag) {
                printf("%d ", i);
        }
}
```

Analizziamo il caso `i=3`

- `j=2; j*j <= i` → `4 <=3` → `false`
- `3` quindi è primo, non lo stampa, perché?

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 5: 2
Primo ciclo i=3

Primo ciclo i=5
Secondo ciclo i,j=5,2
```

Trova l'errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2) {
        flag = 1;
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }
        printf("flag=%d\n",flag);
        if (flag) {
                printf("%d ", i);
        }
}
```

Analizziamo il caso `i=3`

- `j=2; j*j <= i` → `4 <=3` → `false`
- `3` quindi è primo, non lo stampa, perché?
- `flag` non viene gestita correttamente, secondo errore trovato!
- Fino a qui tutto ok, aumento il valore di `n`

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 5: 2
Primo ciclo i=3
3
Primo ciclo i=5
Secondo ciclo i,j=5,2
5
```

Trova l'errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2) {
        flag = 1;
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                if (n % j == 0)
                        flag = 0;
        }
        printf("flag=%d\n",flag);
        if (flag) {
                printf("%d ", i);
        }
}
```

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 10: 2
Primo ciclo i=3
3
Primo ciclo i=5
Secondo ciclo i,j=5,2

Primo ciclo i=7
Secondo ciclo i,j=7,2

Primo ciclo i=9
Secondo ciclo i,j=9,2
Secondo ciclo i,j=9,3
```

Aumentando `n`, anche il caso `n=5` smette di funzionare, il flusso è corretto ma qualcosa non torna ancora

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 5: 2
Primo ciclo i=3
3
Primo ciclo i=5
Secondo ciclo i,j=5,2
5
```

Step 3: verifico le espressioni che controllano iterazioni e selezioni nel programma

Trova l'errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2) {
        flag = 1;
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                printf("n,j,n%%j:%d,%d,%d\n",n,j,n%j);
                if (n % j == 0)
                        flag = 0;
        }
        printf("flag=%d\n",flag);
        if (flag) {
                printf("%d ", i);
        }
}
```

Step 3: verifico le espressioni che controllano iterazioni e selezioni nel programma

Doppio %% per stampare un %

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 10: 2
Primo ciclo i=3
3
Primo ciclo i=5
Secondo ciclo i,j=5,2
n,j,n%j:10,2,0

Primo ciclo i=7
Secondo ciclo i,j=7,2
n,j,n%j:10,2,0

Primo ciclo i=9
Secondo ciclo i,j=9,2
n,j,n%j:10,2,0
Secondo ciclo i,j=9,3
n,j,n%j:10,3,1
```

5 e 7 sono primi, ma `flag` viene settato a 0 perché `10%2 == 0`, ho trovato il terzo errore!

Trova l'errore - debug:

```
printf("Numeri primi fino a %d: 2 ", n);
for (i = 3; i <= n; i+=2) {
        flag = 1;
        printf("\nPrimo ciclo i=%d\n",i);
        for (j = 2; j * j <= i; j++) {
                printf("Secondo ciclo i,j=%d,%d\n",i,j);
                printf("i,j,i%%j:%d,%d,%d\n",i,j,i%j);
                if (i % j == 0)
                        flag = 0;
        }
        printf("flag=%d\n",flag);
        if (flag) {
                printf("%d ", i);
        }
}
```

Step 3: verifico le espressioni che controllano iterazioni e selezioni nel programma

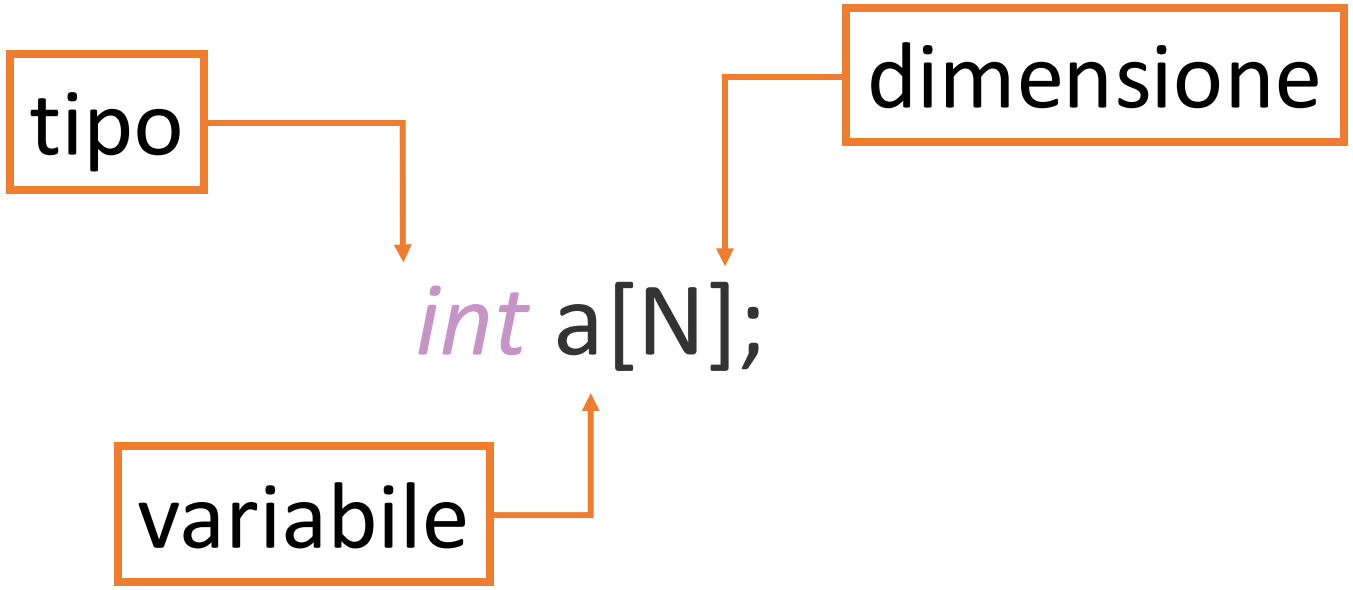
Doppio %% per stampare un %

Trovata la condizione corretta per settare il valore di `flag`, il programma funziona; lo verifico con i casi dei due numeri primi (5,7) e di quello non primo (9).
Rimuovo le stringhe di debug, e posso testare tutto.

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
Numeri primi fino a 10: 2
Primo ciclo i=3
3
Primo ciclo i=5
Secondo ciclo i,j=5,2
i,j,i%j:5,2,1
5
Primo ciclo i=7
Secondo ciclo i,j=7,2
i,j,i%j:7,2,1
7
Primo ciclo i=9
Secondo ciclo i,j=9,2
i,j,i%j:9,2,1
Secondo ciclo i,j=9,3
i,j,i%j:9,3,0
```

```
Inserisci una cifra massima per calcolare i numeri primi: 55
Numeri primi fino a 55: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
```

Vettori / Array in C



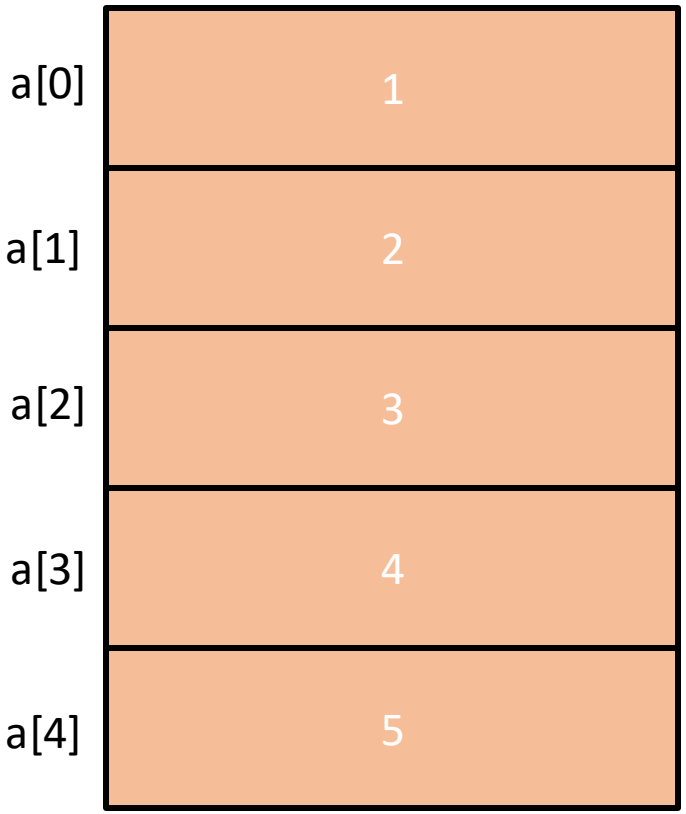
- Struttura contenente un dato numero di oggetti dello stesso tipo;
- Dimensione nota a priori (a compile time)
  - Usare una `define` per la dimensione
- Deve essere inizializzato prima dell'uso
- In memoria allocato sequenzialmente

Vettori / Array in C

- Accesso posizionale – elemento per elemento
- Convenzione: si conta da `0` a `N-1`
- L'operatore `arr[i]` permette di accedere al valore memorizzato all'`i`-esima posizione dell'array `arr`
- **Non c'è controllo se condizione rispettata `i>0 i<N`**
  - Fonte di errori e problemi di sicurezza
- Non è possible usare `=` per assegnare il valore di un array ad un'altra variabile array
  - Possiamo farlo solo con i tipi che abbiamo visto finora

Vettori / Array in C

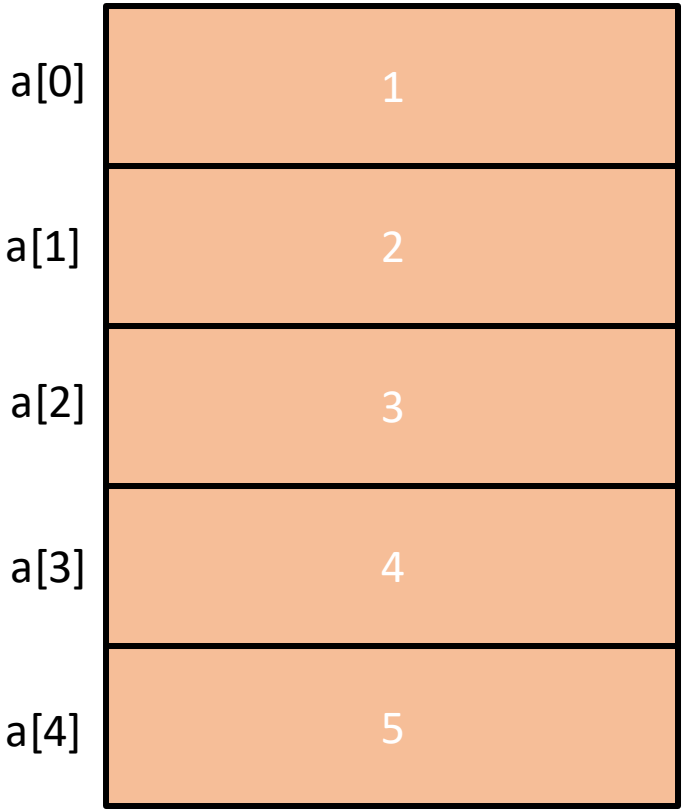
```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        int a[N] = {1,2,3,4,5};

        for (int i=0; i<N; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;

}
```



- Accesso posizionale – elemento per elemento
- Convenzione: si conta da `0` a `N-1`

Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        int a[N];

        for (int i=0; i<N; i++){
                printf("Inserisci %d-esimo elemento: \n",i);
                scanf("%d",&a[i]);
        }

        for (int i=0; i<N; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;
}
```

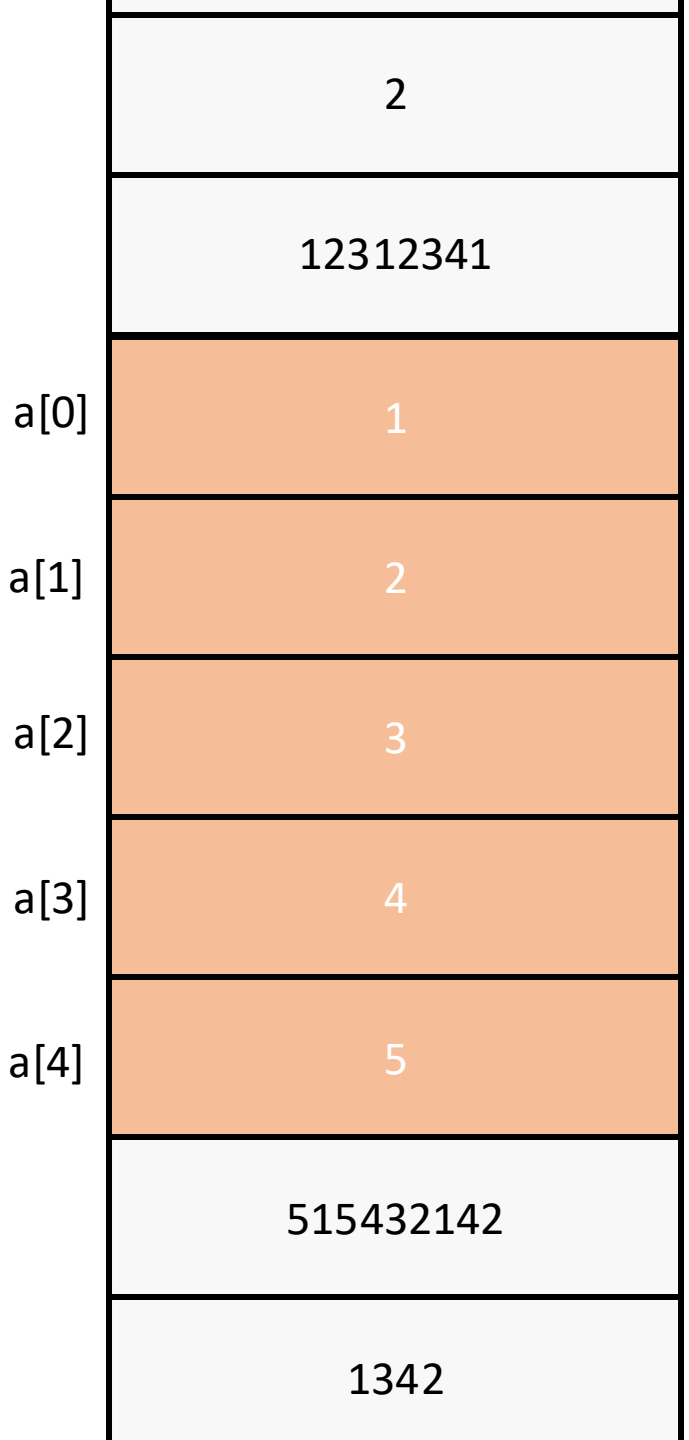


- Accesso posizionale – elemento per elemento
- Convenzione: si conta da `0` a `N-1`

Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){
        int a[N] = {1,2,3,4,5};

        for (int i=0; i<=N; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;
}
```

Errore!



Cosa succede se accedo all'area di memoria prima e dopo l'array?

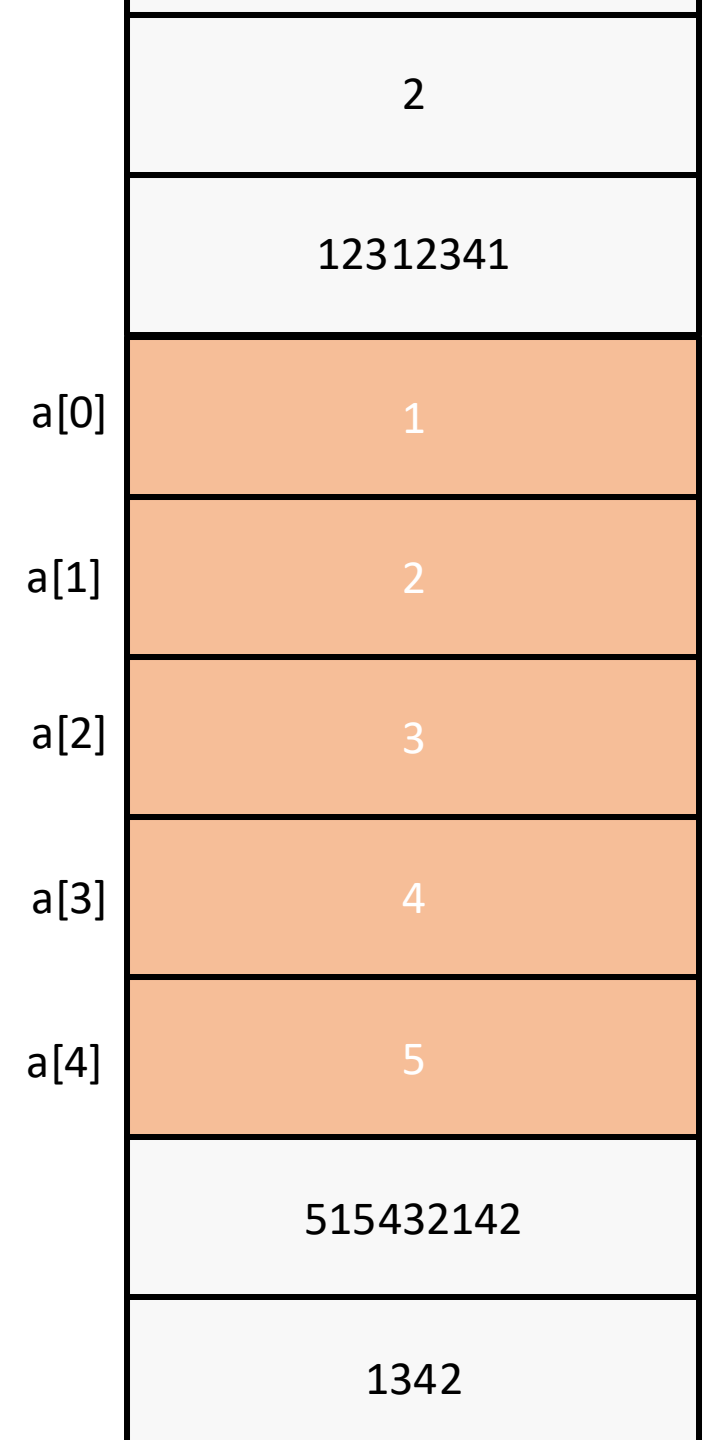
C non controlla che gli intervalli siano corretti

Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){
        int a[N] = {1,2,3,4,5};

        for (int i=0; i<=N; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;
}
```

Errore!

```
goldleaf@GoldBookPro2 L4 % ./a.out
0-esimo elemento: 1
1-esimo elemento: 2
2-esimo elemento: 3
3-esimo elemento: 4
4-esimo elemento: 5
5-esimo elemento: 1
```

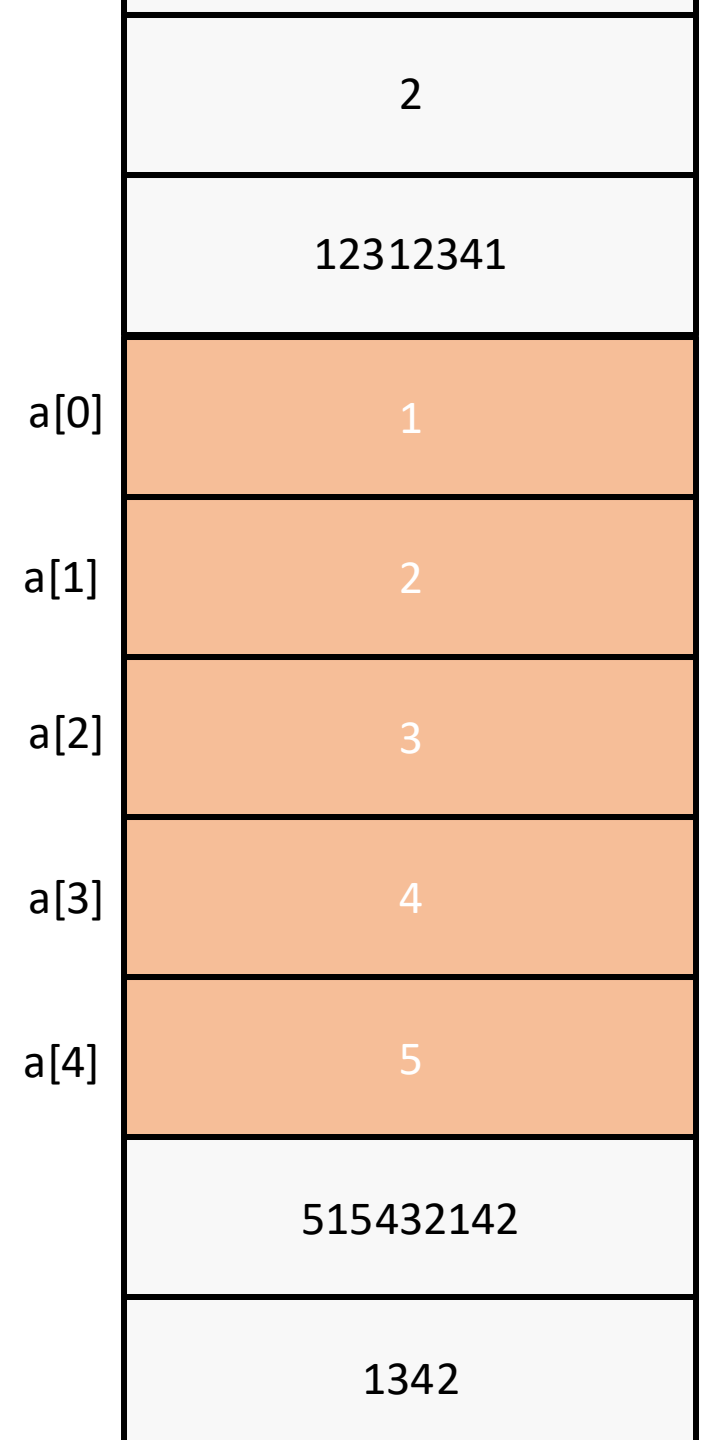


Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){
        int a[N] = {1,2,3,4,5};

        for (int i=0; i<=N; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;
}
```

Errore!

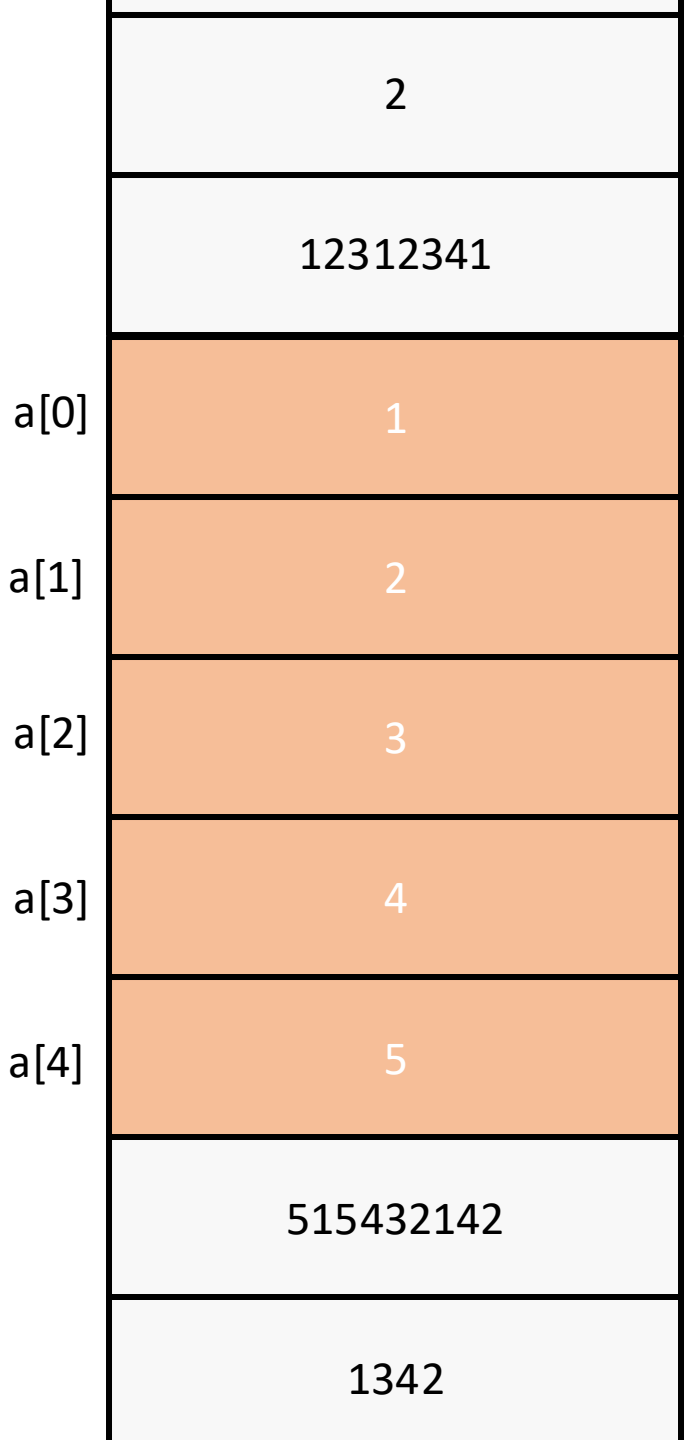
```
goldleaf@GoldBookPro2 L4 % ./a.out
0-esimo elemento: 1
1-esimo elemento: 2
2-esimo elemento: 3
3-esimo elemento: 4
4-esimo elemento: 5
5-esimo elemento: 1
```



Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){
        int a[N] = {1,2,3,4,5};

        for (int i=0; i<=N; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;
}
```

Errore!



Comportamento NON NOTO a compile time = errore a run time

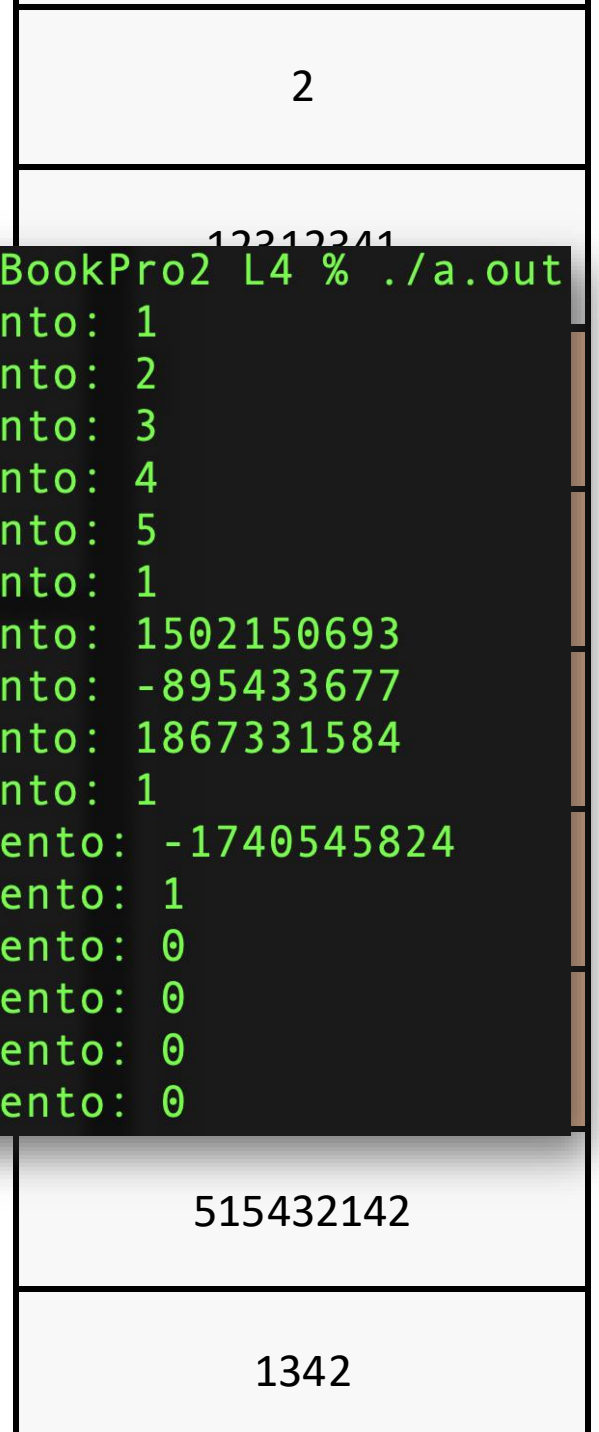
- Caso 1: prima (o dopo) l'array c'è un segmento di memoria accessibile;
  - Il programma continua
  - Sto usando dei valori su cui non ho controllo
- Caso 2: prima (o dopo) l'array c'è un segmento di memoria NON accessibile;
  - Il programma viene terminato
  - Errore a runtime (Segmentation Fault)

Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

int a[N] = {1,2,3,4,5};

for (int i=0; i<=N+10; i++)
        printf("%d-esimo elemento: %d\n",i,a[i]);

return 0;

}
```

Errore!

```
goldleaf@GoldBookPro2 L4 % ./a.out
0-esimo elemento: 1
1-esimo elemento: 2
2-esimo elemento: 3
3-esimo elemento: 4
4-esimo elemento: 5
5-esimo elemento: 1
6-esimo elemento: 1502150693
7-esimo elemento: -895433677
8-esimo elemento: 1867331584
9-esimo elemento: 1
10-esimo elemento: -1740545824
11-esimo elemento: 1
12-esimo elemento: 0
13-esimo elemento: 0
14-esimo elemento: 0
15-esimo elemento: 0
```



Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

int a[N] = {1,2,3,4,5};

for (int i=0; i<=N+1000; i++)
        printf("%d-esimo elemento: %d\n",i,a[i]);

return 0;

}
```

Errore!

```
629-esimo elemento: 1698194739
630-esimo elemento: 1650865462
631-esimo elemento: 1697724210
632-esimo elemento: 959527223
633-esimo elemento: 1630942819
634-esimo elemento: 878862903
635-esimo elemento: 1647797560
636-esimo elemento: 2019885110
637-esimo elemento: 1953850213
638-esimo elemento: 1701601889
639-esimo elemento: 1869570655
640-esimo elemento: 1935763572
641-esimo elemento: 912473448
642-esimo elemento: 808542776
643-esimo elemento: 1667445297
644-esimo elemento: 959985974
645-esimo elemento: 1714698290
646-esimo elemento: 858863203
647-esimo elemento: 859071842
648-esimo elemento: 942761572
649-esimo elemento: 1701197158
650-esimo elemento: 862138677
651-esimo elemento: 1627402290
652-esimo elemento: 875982194
653-esimo elemento: 1650548581
654-esimo elemento: 1936670057
655-esimo elemento: 1600680960
656-esimo elemento: 1953656688
657-esimo elemento: 61
658-esimo elemento: 0
659-esimo elemento: 0
zsh: segmentation fault  ./a.out
goldleaf@GoldBookPro2 L4 %
```

Vettori / Array in C

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        int a[N] = {1,2,3,4,5};
        int b[N];

        b = a;

        for (int i=0; i<N; i++)
                printf("%d-esimo elemento di a:          %d\n",i,a[i]);

        for (int i=0; i<N; i++)
                printf("%d-esimo elemento di b: %d\n",i,a[i]);

        return 0;

}
```

Errore!

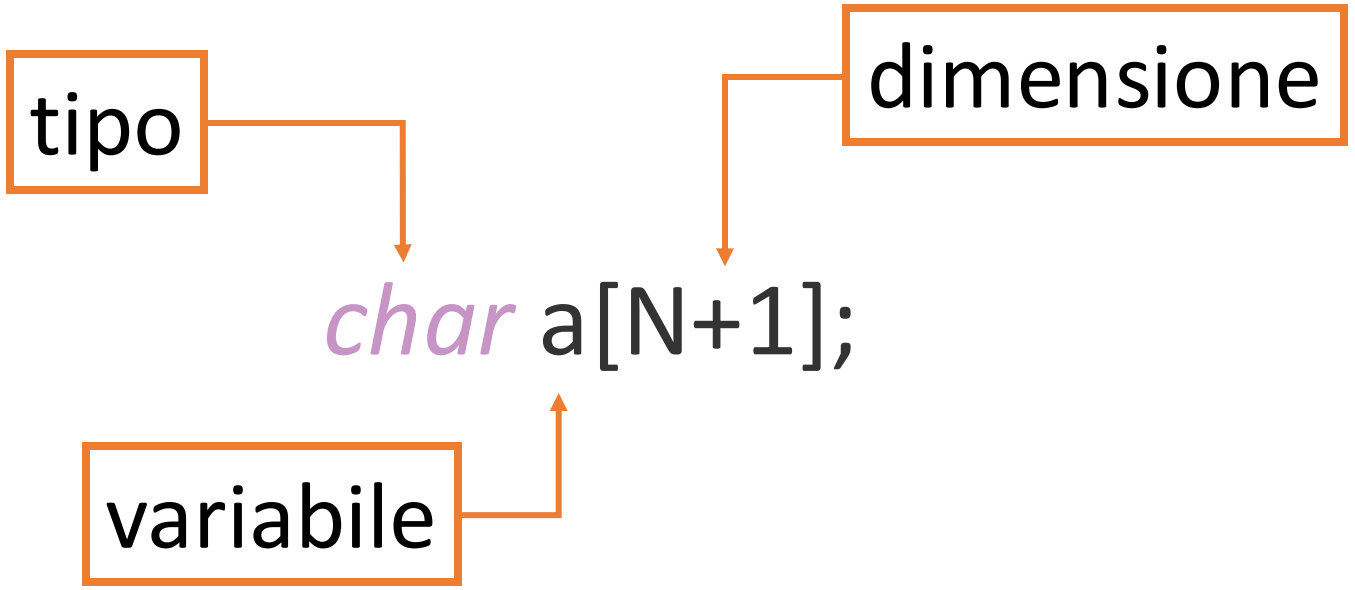
```
goldleaf@GoldBookPro2 L4 % gcc errorecopia.c
errorecopia.c:10:4: error: array type 'int[5]' is not assignable
        b = a;
        ~ ^
1 error generated.
goldleaf@GoldBookPro2 L4 %
```

Non posso copiare direttamente un array in un altro in una sola operazione

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        int a[N] = {1,2,3,4,5};
        int b[N];
        int i;

        for (i=0; i<N; i++)
                b[i] = a[i];

        for (i=0; i<N; i++)
                printf("%d-esimo elemento di a: %d\n",i,a[i]);

        for (i=0; i<N; i++)
                printf("%d-esimo elemento di b: %d\n",i,b[i]);

        return 0;
}
```

Ma devo copiare elemento per elemento

Stringhe



- Una stringa è un array di `char`
- Dimensione nota a priori (a compile time)
  - Usare una `define` per la dimensione
  - Dimensione: lunghezza massima stringa + 1 (terminatore)
  - Dimensione massima (per convenzione): costante `BUFSIZ` (1024)

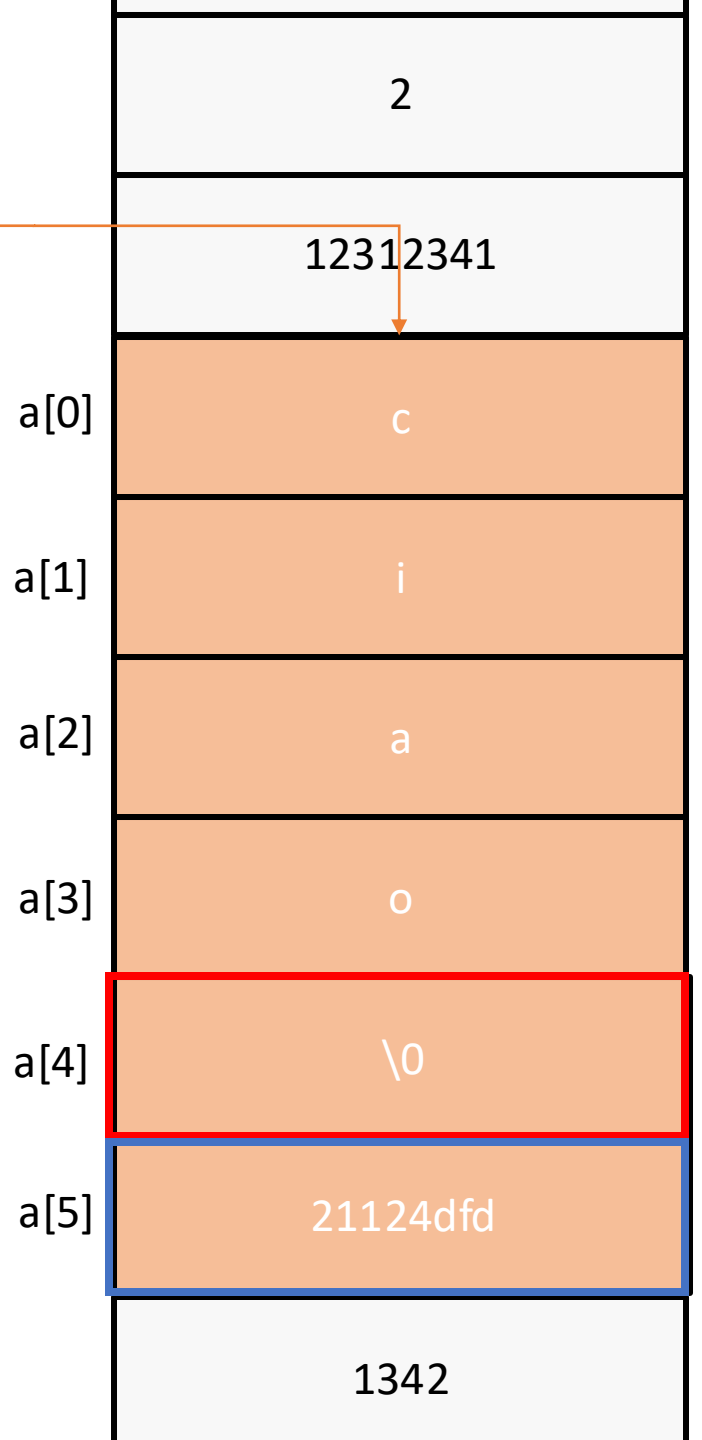
Stringhe

```
#define N 5

int main(){
        char a[N+1] = "ciao";
        return 0;
}
```

Porzione di memoria riservata al momento della dichiarazione (`N+1` celle dimensionate per contenere `char`)

Terminatore (aggiunto automaticamente quando inizializziamo una stringa nel modo mostrato)

Il valore di questa cella di memoria non è stato inizializzato, non sappiamo cosa contiene



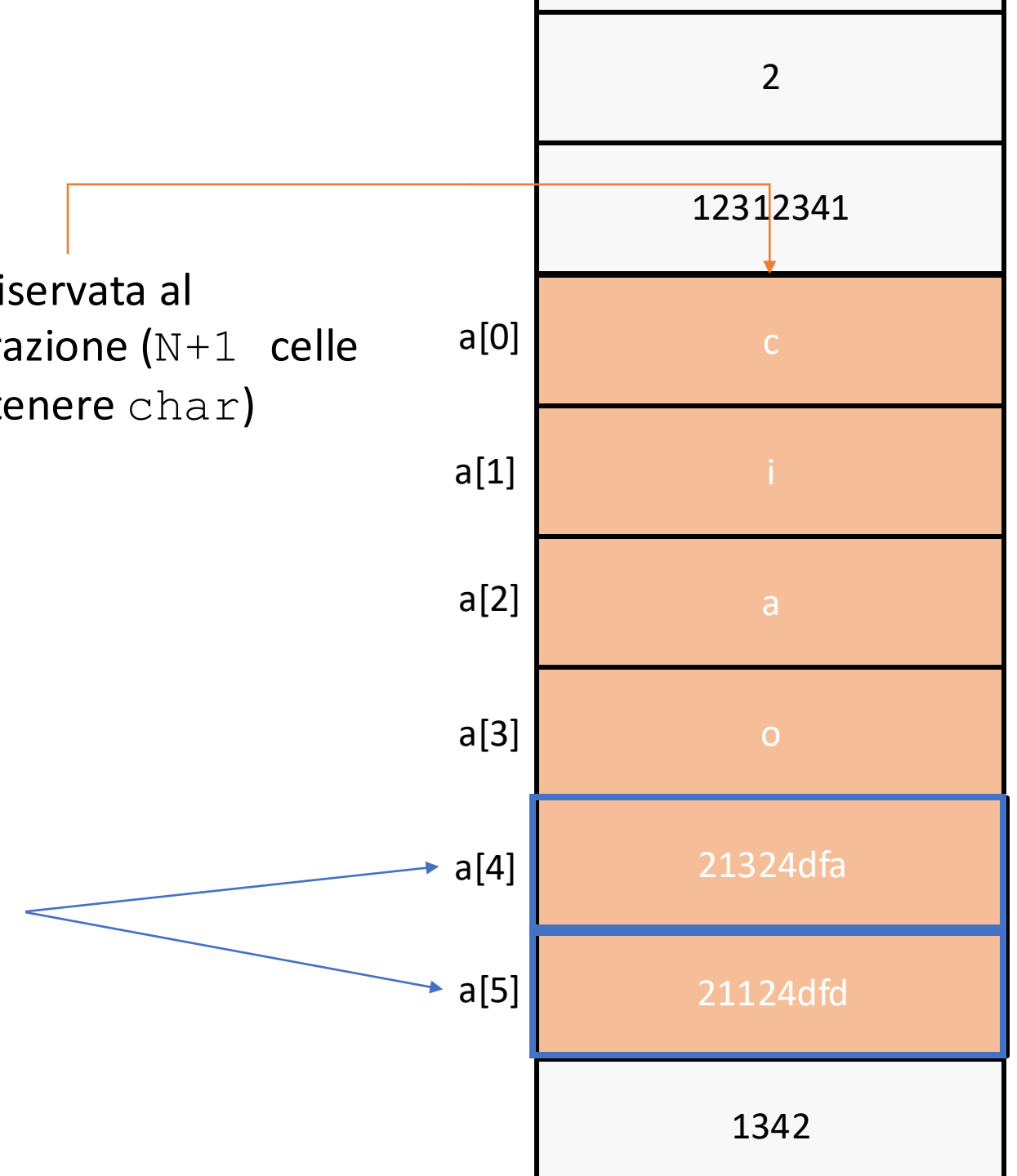
Stringhe

```
#define N 5

int main(){

        char a[N+1] = {'c', 'i', 'a', 'o'};
        return 0;

}
```

Porzione di memoria riservata al momento della dichiarazione (`N+1` celle dimensionate per contenere `char`)

Il valore di queste celle di memoria non è stato inizializzato, non sappiamo cosa contengono

Il <u>terminatore non viene aggiunto automaticamente</u> se dichiariamo la stringa come un normale array



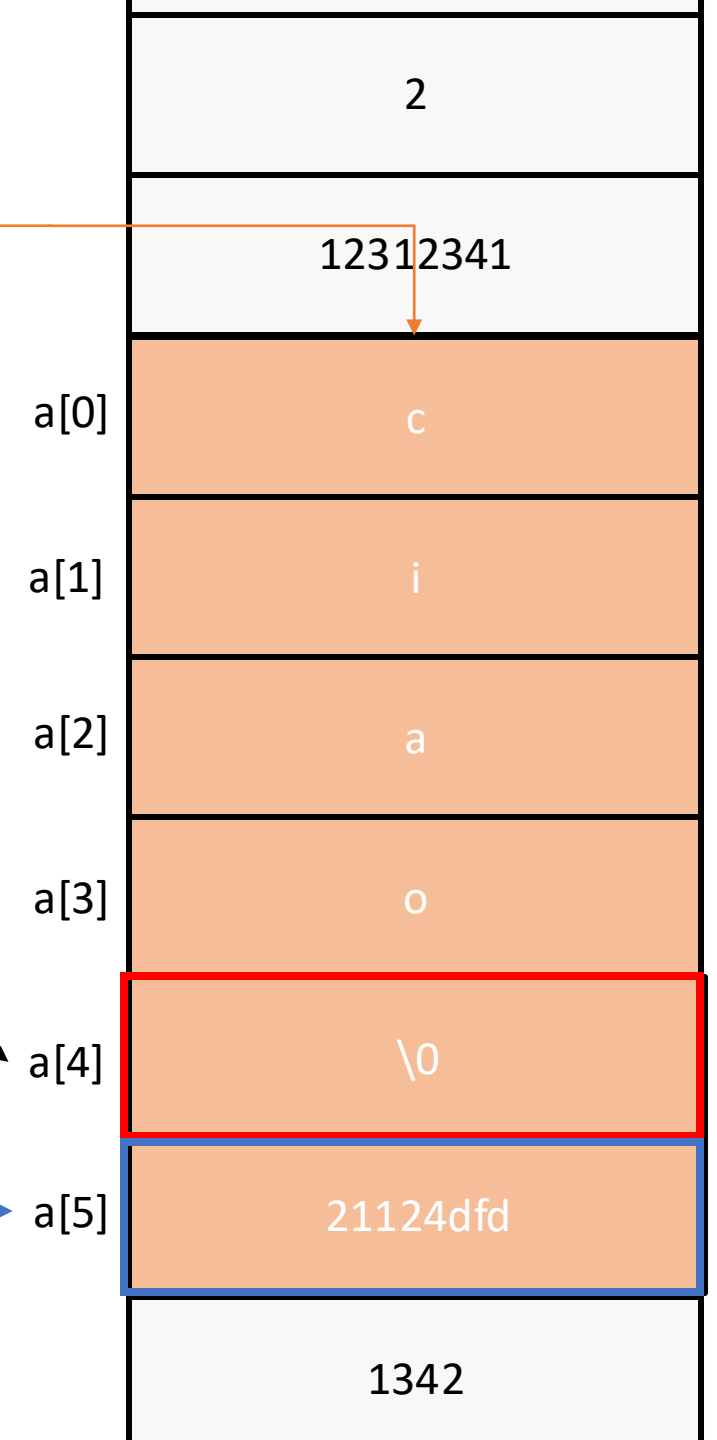
Stringhe

```
#define N 5

int main(){

        char a[N+1] = {'c ', ' i ', ' a ', ' o ' , ' \0 '};
        return 0;

}
```

Porzione di memoria riservata al momento della dichiarazione (`N+1` celle dimensionate per contenere `char`)

Terminatore aggiunto manualmente

Il valore di questa cella di memoria non è stato inizializzato, non sappiamo cosa contiene



Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `scanf` non dobbiamo usare `&` (poi capiremo perché)
  - Il terminatore viene aggiunto automaticamente

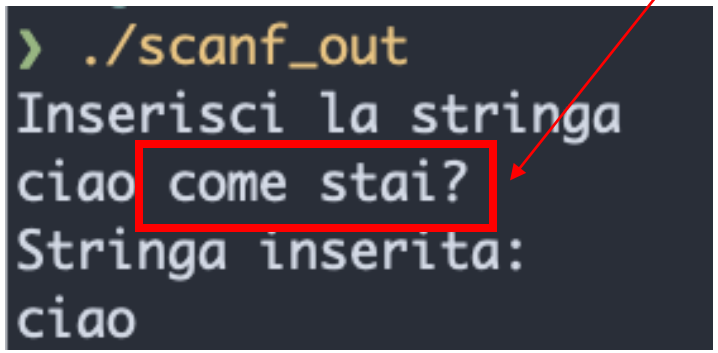
```c
char a[N+1];
printf("Inserisci la stringa\n");
scanf("%s",a);
printf("Stringa inserita:\n%s\n",a);
```

```
❯ ./scanf_out
Inserisci la stringa
ciao
Stringa inserita:
ciao
```

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `scanf` non dobbiamo usare `&` (poi capiremo perché)
  - Il terminatore viene aggiunto automaticamente
  - La lettura della stringa termina al primo spazio bianco

```
char a[N+1];
printf("Inserisci la stringa\n");
scanf("%s",a);
printf("Stringa inserita:\n%s\n",a);
```

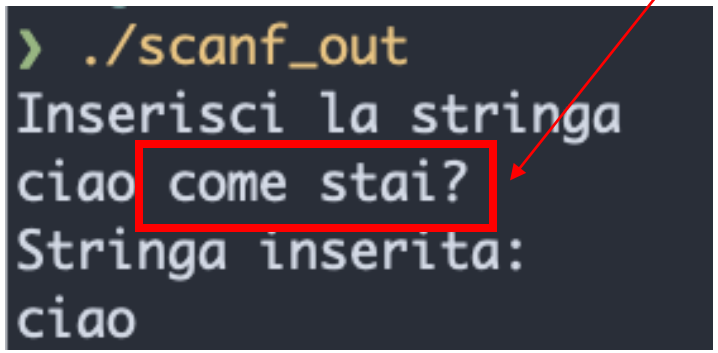


Solo `ciao` fa match con `%s`, `come stai` non fa match

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `scanf` non dobbiamo usare `&` (poi capiremo perché)
  - Il terminatore viene aggiunto automaticamente
  - La lettura della stringa termina al primo spazio bianco

```
char a[N+1];
printf("Inserisci la stringa\n");
scanf("%s",a);
printf("Stringa inserita:\n%s\n",a);
```



Solo `ciao` fa match con `%s`, `come stai` non fa match

Se volessimo leggere una stringa di più parole?

Leggere Stringhe di più parole

```
char a[N+1];
printf("Inserisci la stringa\n");
fgets(a, sizeof(a), stdin);
printf("Stringa inserita:\n%s\n",a);
```

```
goldleaf@GoldBookPro2 ESEMPI % ./stringaLunga
Inserisci la stringa
ciao sono stringa!
Stringa inserita:
ciao sono stringa!

goldleaf@GoldBookPro2 ESEMPI %
```

Leggere Stringhe di più parole

fgets(stringa, sizeof(stringa), stdin);

Variabile dove mettere la stringa letta

Numero max di caratteri che possono essere letti (evita overflow, che è un problema della *gets*)

Da dove leggo la stringa, in questo caso da stdin, ma più avanti anche da file (prox lezioni)

Leggere Stringhe di più parole

```
char a[N+1];
printf("Inserisci la stringa\n");
fgets(a, sizeof(a), stdin);
printf("Stringa inserita:\n%s\n",a);
```

fgets legge anche il carattere di newline \n inserito dall'utente;
Se lo volete togliere, va rimosso manualmente

```
str[strcspn(str, "\n")] = '\0';
```

```
[goldleaf@GoldBookPro2 ESEMPI % ./stringaLunga
[Inserisci la stringa
 ciao sono stringa!
 Stringa inserita:
 ciao sono stringa!

 goldleaf@GoldBookPro2 ESEMPI %
```

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `scanf` non dobbiamo usare `&` (poi capiremo perché)
  - Il terminatore viene aggiunto automaticamente
  - La lettura della stringa termina al primo spazio bianco
- Non c'è controllo che ci sia spazio sufficiente

```
#define N 4

char a[N+1];
printf("Inserisci la stringa\n");
scanf("%s",a);
printf("Stringa inserita:\n%s\n",a);
```

```
❯ ./scanf_out
Inserisci la stringa
ciao,comestai?????
Stringa inserita:
ciao,comestai?????
fish: Job 1, './scanf_out' terminated by signal SIGBUS (Misaligned address error)
```

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `scanf` non dobbiamo usare `&` (poi capiremo perché)
  - Il terminatore viene aggiunto automaticamente
  - La lettura della stringa termina al primo spazio bianco
- Non c'è controllo che ci sia spazio sufficiente

```
#define N 4

char a[N+1];
printf("Inserisci la stringa\n");
scanf("%s",a);
printf("Stringa inserita:\n%s\n",a);
```

```
❯ ./scanf_out
Inserisci la stringa
ciao come stai?
Stringa inserita:
ciao
```

Come mai qui non succede nulla di strano?

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `printf` la stringa viene stampata fino al (primo) terminatore

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella `printf` la stringa viene stampata fino al (primo) terminatore

```
char a[N+1] = {'c', 'i', 'a', 'o', '\0 ', '!'};
printf("Stringa:\n%s\n",a);
```

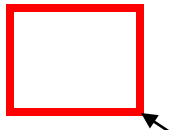
```
> ./printf_corner_case
La stringa e': ciao
```

Stringhe

- Placeholder `%s` per leggere/scrivere una stringa
- Nella printf la stringa viene stampata fino al (primo) terminatore

```
char a[N+1] = {'c', 'i', 'a', 'o', '\0 ', '!'};
printf("Stringa:\n%s\n",a);
```

```
> ./printf_corner_case
La stringa e': ciao
```



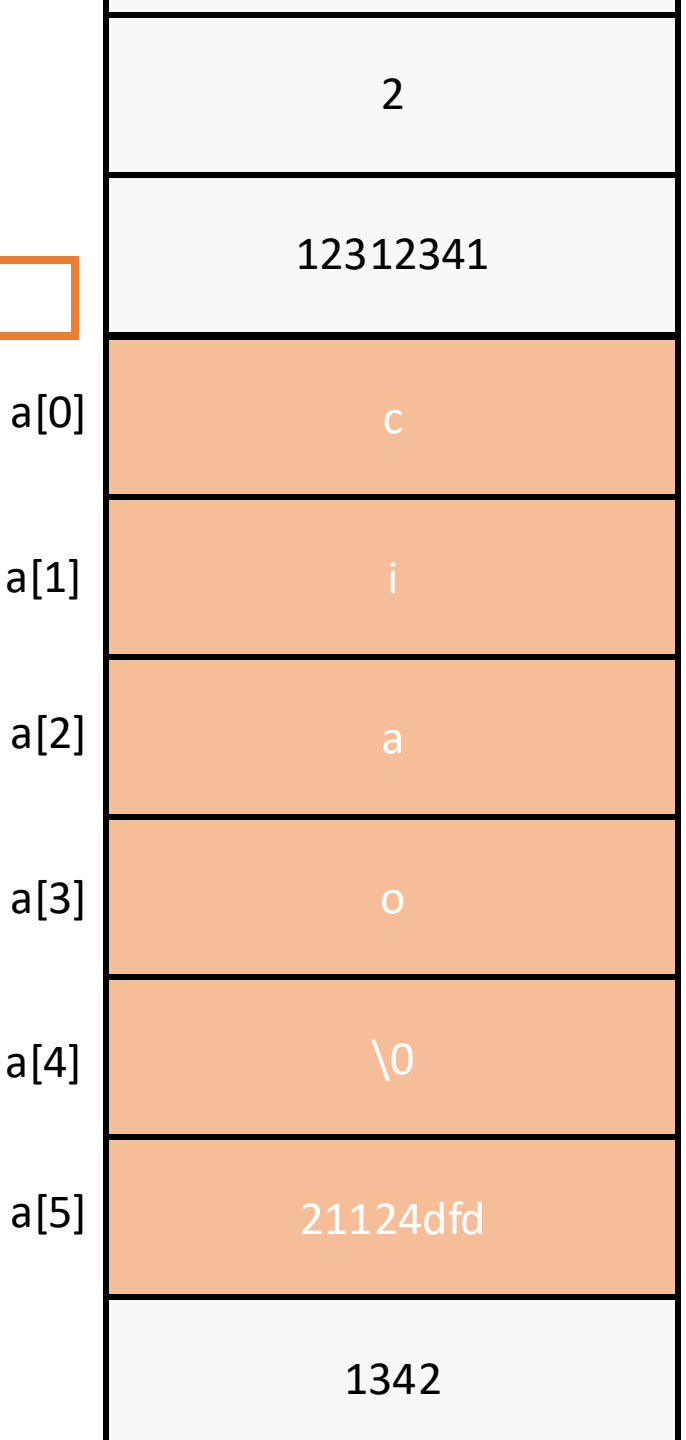
! non viene mostrato perché è dopo il terminatore

Stringhe

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        char a[N+1];
        printf("Dimensione max stringa%d\n",BUFSIZ);

        printf("Inserisci la stringa\n");
        scanf("%s",a);
        printf("Stringa inserita:\n%s\n",a);

        return 0;

}
```

+1 elemento: terminatore

senza la &



- Posso leggere e stampare le stringhe con una sola istruzione o posizionalmente
- La stringa viene stampata fino al terminatore

Lo standard input `stdin`

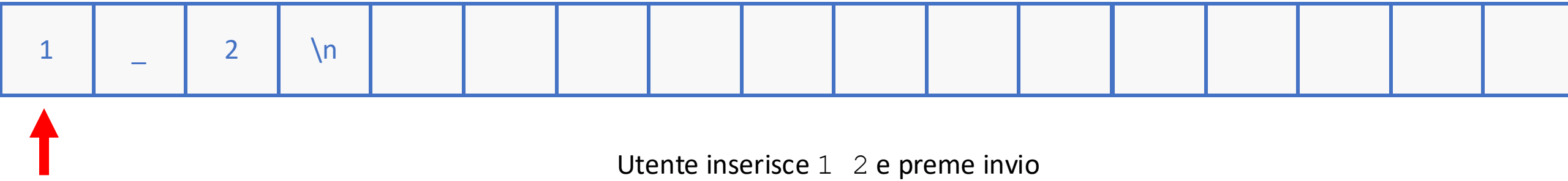
- Funzionamento: simile ad un «nastro» di caratteri con una «testina» mobile che si sposta con le letture
- Ogni lettura «consuma» l'input letto, lascia il resto sul buffer
- Se il buffer è vuoto, aspetta input da utente
- Se il buffer è pieno, legge quello che contiene e vede se corrisponde alle variabili da leggere
  - Se corrisponde, aggiorna la variabile
  - Se non corrisponde, consuma il dato e va avanti

Lo standard input `stdin`

```
scanf("%d",&x);
scanf("%d",&y);
printf("%d %d\n",x,y);
```

Buffer vuoto, il terminale lampeggia e attende input da utente

Lo standard input `stdin`



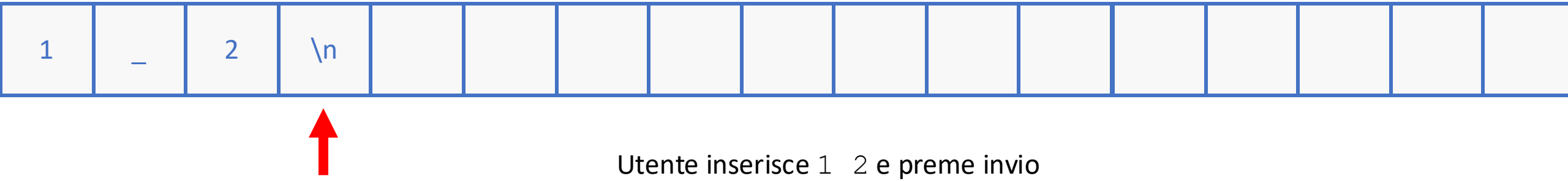
Utente inserisce `1  2` e preme invio

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
1 2
1 2
```

```
scanf("%d",&x);
scanf("%d",&y);
printf("%d %d\n",x,y);
```

La prima `scanf` legge il primo valore e lo mette in `x`

Lo standard input `stdin`




```
scanf("%d",&x);
scanf("%d",&y);
printf("%d %d\n",x,y);
```

Utente inserisce `1  2` e preme invio

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
1 2
1 2
```

La prima `scanf` legge il primo valore e lo mette in `x`

La seconda `scanf` legge lo spazio, e lo scarta (non è `int`)

Il buffer non è vuoto, va avanti e legge il secondo valore, lo mette in `y`

Sullo `stdin` rimane il valore `\n`

Lo standard input `stdin`


```
int x;
char str[N];

scanf("%d",&x);
fgets(str,sizeof(str),stdin);
printf("char vale: %d.\nstringa vale: %s.\n",x,str);
```

```
goldleaf@GoldBookPro2 ESEMPI % ./a.out
12
char vale: 12.
stringa vale:
.
```

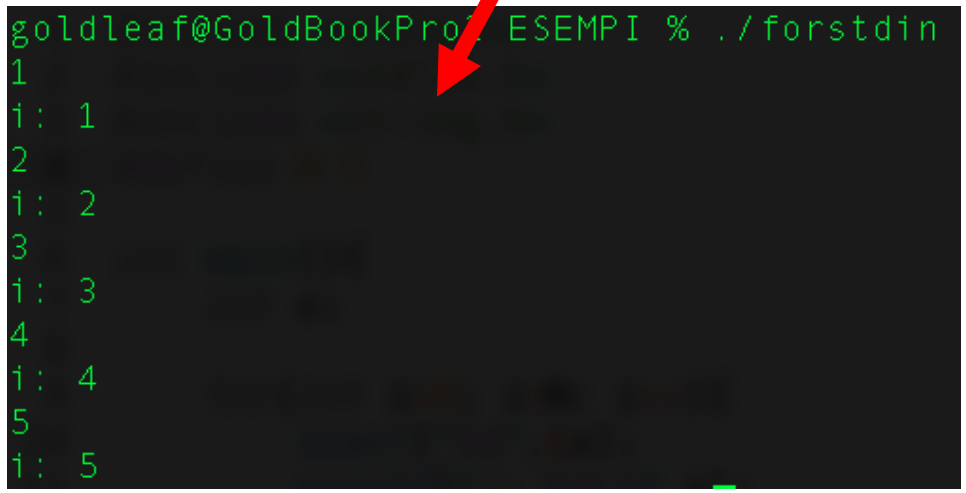
Utente inserisce `12` e preme invio, la `scanf` consuma il `12` ma lascia il `\n`; la `fgets` successiva legge il `\n` e termina. `str` vale `\n`

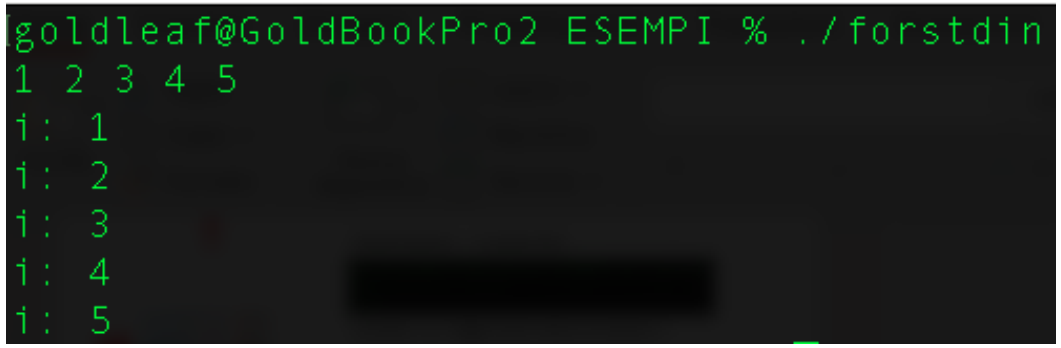
Lo standard input `stdin`

```
for(int i=0; i<N; i++){
        scanf("%d",&x);
        printf("i: %d\n",x);
}
```

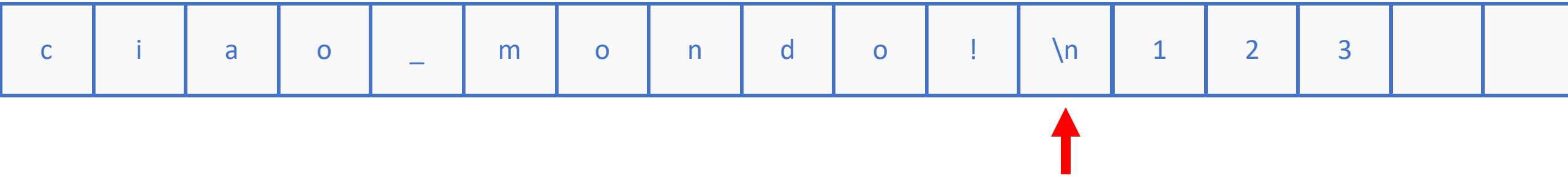
1	_	2	_	3	_	4	_	5	_	c	i	a	o	\n		

Potete «riempire» lo standard input in un colpo solo;
Esempio: se volete dare `N` valori per in un `for` potete dare `N` valori inseriti singolarmente o tutti assieme





Lo standard input `stdin`



- Diverse letture fatte sequenzialmente di `scanf getchar fgets` possono lasciare qualcosa sullo `stdin` che può venir letto in un momento successivo dal programma (anche molto successivo)
- Se il vostro programma funziona in maniera anomala, controllate cosa avete letto per evitare questi tipi di errori.
- In `C` non c'è una soluzione unica per avere un input robusto ad errori degli utenti (non serve di fatto se non per esempi didattici), accontentatevi del fatto che il vostro programma sia robusto *abbastanza.*
  Soluzioni più complicate sono un *overkill* per gli esercizi richiesti durante questo corso, non è quello il senso degli esercizi proposti.

Variable-Length Array

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        int n;
        printf("Quando vuoi grande l'array?\n");
        scanf("%d",&n);
        int a[n];

        for (int i=0; i<n; i++){
                printf("Inserisci %d-esimo elemento: \n",i);
                scanf("%d",&a[i]);
        }

        for (int i=0; i<n; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;

}
```

- C99 permette di dichiarare array di dimensione nota solo a run time, i VLA
- Sebbene alcuni compilatori li supportano, non sono una pratica incoraggiata nel C
- In questo corso non potete usarli (vedremo poi come allocare dinamicamente la memoria)

Variable-Length Array

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5

int main(){

        int n;
        printf("Quando vuoi grande l'array?
        scanf("%d",&n);
        int a[n];

        for (int i=0; i<n; i++){
                printf("Inserisci %d-esimo elemento: \n",i);
                scanf("%d",&a[i]);
        }

        for (int i=0; i<n; i++)
                printf("%d-esimo elemento: %d\n",i,a[i]);

        return 0;

}
```

> Secondly, the compiler is usually very stupid, and will generate horrible code for VLA's.
>
> Third, there's no guarantee that the compiler will actually even realize that the size is limited, and guarantee that it won't screw up the stack.
>
> So no. VLA's are not acceptable in the kernel. Don't do them. We're getting rid of them.
>
> Linus

Durante l'esame: usare VLA = errore

Array e Variable-Lenght Array

```
/* non usare VLA */
int n;
scanf("%d",&n);
int a[n];
for(i=0;i<n;i++)
     scanf("%d",&a[i]);
```

```
/* usate le define*/
#define N 100 // >> max size
…
int n,a[N];
do {
     scanf("%d",&n);}
while (n>N);

for(i=0;i<n;i++)
     scanf("%d",&a[i]);
```

Invece di usare i VLA, sovradimensionate la dimensione degli array con una define, e usate i primi n<N elementi (come con le stringhe con `BUFSIZ`).

Trova l’errore:

```c
#include <stdio.h>
/* Scrivere un programma che prende in input due numeri interi,
k, e n, e trova il massimo numero divisore di k e minore di n */
int main(){

    int n;
    int k;
    int i;
    int result = -1; // valore fuori dal dominio
    do {
        printf("Inserisci n e k separati da spazio: ");
        scanf("%d %d", &n, &k);
    } while (n <= 0 && k <= 0);

    for (i = 0; i < n; i++){
        if (i % k == 0) {
            result = i;
        }
    }
    if (result == -1) {
        printf("Non ho trovato nessun divisore.\n");
    } else {
        printf("Il massimo divisore: %d\n", result);
    }

    return 0;
}
```