

PROGRAMMAZIONE 2: SPERIMENTAZIONI

Lezione 6 (parte 2) – Algoritmi ricorsivi in C

Agenda

- Introduzione
 - Potenza di un numero
 - Sommatoria di n numeri
 - Array
 - Visualizzare gli elementi di un array
 - Sommare gli elementi di un array
 - Cercare un valore nell'array
 - Cercare il valore massimo nell'array
 - Visualizzare una stringa
 - Lunghezza di una stringa
 - Contare le maiuscole in una stringa
 - Trasformare una stringa



Video lezione disponibile su YouTube: <https://youtu.be/r6VA1HSVGRs>


Introduzione

- In questa lezione verranno presentati alcuni algoritmi ricorsivi utilizzando dapprima le variabili standard del linguaggio C quindi array numerici e stringhe (array di caratteri).
- **Prerequisiti:** lezione 6 (parte 1).

Potenza di un numero

- **Caso base:**
 - l'esponente è 0 quindi il risultato è 1.
- **Caso ricorsivo:**
 - base * potenza(esponente-1).

$$f(x, y) = \begin{cases} 1, & y = 0 \\ x \times f(x, y - 1), & y > 0 \\ \frac{1}{f(x, -y)}, & y < 0 \end{cases}$$



In questo esempio è stato esaminato solo il caso con esponente maggiore di 0.

Potenza di un numero

```
28  double potenza(double base, int expo)
29  {
30      // caso base
31      if (expo == 0)
32          return 1;
33      // caso ricorsivo
34      if (expo > 0)
35          return base * potenza(base, expo - 1);
36  }
```

es1.c

Potenza di un numero

base = 4 expo = 2

2

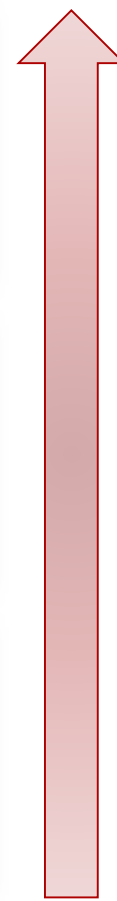
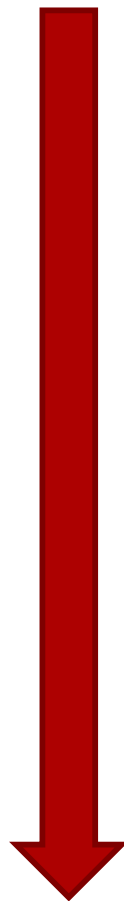
```
28 double potenza(double base, int expo)
29 {
30     // caso base
31     if (expo == 0)
32         return 1;
33     // caso ricorsivo
34     if (expo > 0)
35         return base * potenza(base, expo - 1);
36 }
```

1

```
28 double potenza(double base, int expo)
29 {
30     // caso base
31     if (expo == 0)
32         return 1;
33     // caso ricorsivo
34     if (expo > 0)
35         return base * potenza(base, expo - 1);
36 }
```

0

```
28 double potenza(double base, int expo)
29 {
30     // caso base
31     if (expo == 0)
32         return 1;
33     // caso ricorsivo
34     if (expo > 0)
35         return base * potenza(base, expo - 1);
36 }
```



Sommatoria di n numeri

- **Caso base:**

- il contatore degli elementi da sommare ha raggiunto il limite.

- **Caso ricorsivo:**

- contatore + sommatoria(contatore+1, limite).

$$f(n, x) = \begin{cases} n, & n = x \\ n + f(n + 1, x), & n < x \end{cases}$$

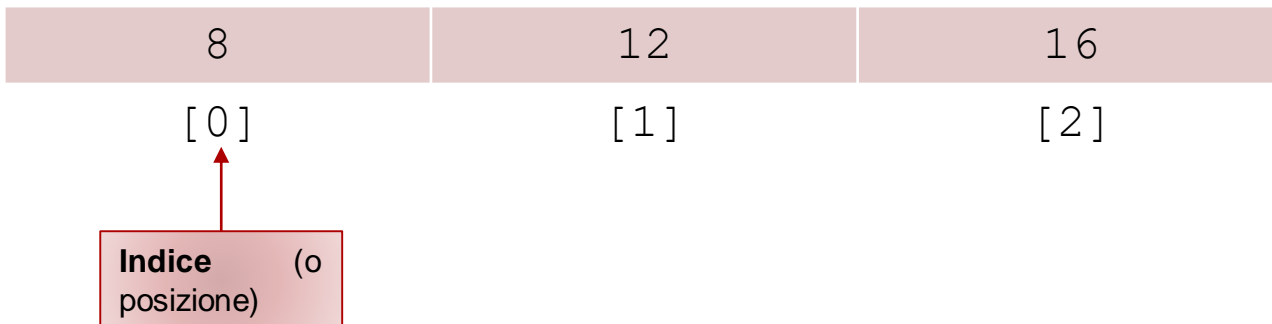
Sommatoria di n numeri

```
25  int sommatoria(int start, int end)
26  {
27      if(start == end)
28          return start;
29      else
30          return start + sommatoria(start + 1, end);
31  }
```

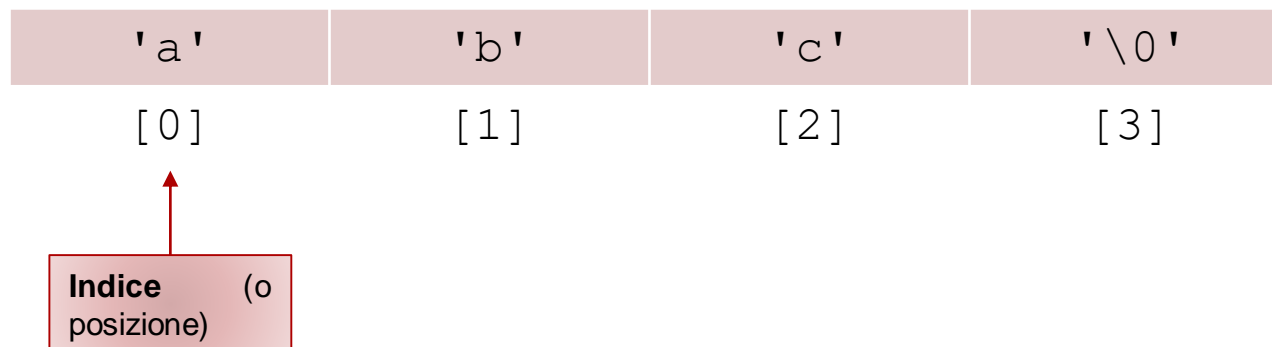
es2.c

Array

- Vengono qui di seguito illustrati alcuni algoritmi ricorsivi sugli array e le stringhe (array di char).



Vettore di interi di lunghezza 3



Stringa (array di char) di lunghezza 3

Visualizzare gli elementi di un array

- **Caso base:**
 - se la posizione (indice) del vettore è maggiore o uguale della dimensione del vettore interrompe la funzione con un `return`.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva del vettore.

Visualizzare gli elementi di un array

```
33 void array_visualizza(int vett[], int p, int len)
34 {
35     // caso base
36     if (p >= len)
37         return;
38
39     printf("[%d]:%d \n", p, vett[p]);
40
41     // caso ricorsivo
42     return array_visualizza(vett, p + 1, len);
43 }
```

es3.c

Sommare gli elementi di un array

- **Caso base:**
 - se la posizione (indice) del vettore è maggiore o uguale della dimensione del vettore restituisce 0.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva del vettore.

Sommare gli elementi di un array

```
32  int array_somma(int vett[], int p, int len)
33  {
34      // caso base
35      if (p >= len)
36          return 0;
37
38      // caso ricorsivo
39      return (vett[p] + array_somma(vett, p + 1, len));
40  }
```

es4.c

Cercare un valore nell'array

- **Caso base:**
 - se la posizione (indice) del vettore è maggiore o uguale della dimensione del vettore restituisce -1.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva del vettore.

Cercare un valore nell'array

```
41  int array_cerca(int vett[], int p, int len, int x)
42  {
43      // caso base
44      if (p >= len)
45          return -1;
46
47      // caso ricorsivo
48      if (vett[p]==x)
49          return p;
50      else
51          return array_cerca(vett, p + 1, len, x);
52  }
```

es5.c

Cercare il massimo nell'array

- **Caso base:**
 - se la posizione (indice) del vettore è maggiore o uguale della dimensione del vettore restituisce -1.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva del vettore.

Cercare il massimo nell'array

es6.c

```

33 void array_max(int vett[], int p, int len, int *r)
34 {
35     // caso base
36     if (p >= len)
37         return;
38
39     // caso ricorsivo
40     if (p==0){ // primo elemento come "campione"
41         *r = vett[0];
42     }
43     else{
44         if (vett[p] > *r){
45             *r = vett[p];
46         }
47     }
48     return array_max(vett, p + 1, len, r);
49 }

```

Visualizzare una stringa

- **Caso base:**
 - se il carattere corrente è `'\0'` interrompe la ricorsione con un `return`.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva della stringa.

Visualizzare una stringa

```

21 void stringa_view(char *s)
22 {
23     // caso base
24     // *s accede al contenuto puntato da s
25     if (*s == '\0')
26         return;
27
28     // s contiene l'indirizzo di memoria puntato
29     printf("Carattere puntato: %p -> valore: %c (%d) \n", s, *s, *s);
30
31     // caso ricorsivo
32     // s+1 sposta il puntatore all'elemento successivo della stringa
33     return stringa_view(s+1);
34 }

```

es7.c

Lunghezza di una stringa

- **Caso base:**
 - se il carattere corrente è ' \0 ' restituisce 0.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva della stringa.

Lunghezza di una stringa

```

24  int stringa_len(char *s)
25  {
26      // caso base
27      // *s accede al contenuto puntato da s
28      if (*s == '\0')
29          return 0;
30
31      // caso ricorsivo
32      // s+1 sposta il puntatore all'elemento successivo della stringa
33      return 1 + stringa_len(s+1);
34  }

```

es8.c

Contare le maiuscole in una stringa

- **Caso base:**
 - se il carattere corrente è `'\0'` interrompe la ricorsione con un `return`.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva della stringa.

Contare le maiuscole in una stringa

```

28 void stringa_maiuscole(char *s, int *count)
29 {
30     // caso base
31     // *s accede al contenuto puntato da s
32     if (*s == '\0')
33         return;
34
35     // caso ricorsivo
36     if (isupper(*s)){
37         *count = *count + 1;
38     }
39
40     return stringa_maiuscole(s+1, count);
41 }

```

es9.c

Trasformare una stringa

- **Caso base:**
 - se il carattere corrente è `'\0'` interrompe la ricorsione con un `return`.
- **Caso ricorsivo:**
 - richiama la funzione sulla posizione (indice) successiva della stringa.

Trasformare una stringa

Versione con puntatore

```

29 void stringa_maiuscole(char *s)
30 {
31     // caso base
32     // *s accede al contenuto puntato da s
33     if (*s == '\0')
34         return;
35
36     // caso ricorsivo
37     if (islower(*s)){
38         *s = toupper(*s);
39     }
40     // s+1 sposta il puntatore all'elemento successivo della stringa
41     stringa_maiuscole(s+1);
42 }

```

es10.c

Trasformare una stringa

Versione con array

```

45 void stringa_maiuscole(char s[], int p)
46 {
47     // caso base
48     if (s[p] == '\0')
49         return;
50
51     // caso ricorsivo
52     if (islower(s[p])){
53         s[p] = toupper(s[p]);
54     }
55
56     // p+1 sposta l'indice all'elemento successivo dell'array
57     return stringa_maiuscole(s, p+1);
58 }

```

es10.c

FINE PRESENTAZIONE

