Funzioni con numero variabile di parametri: Funzioni Variadiche in C

Massimo Benerecetti

Laboratorio di Algoritmi e Strutture Dati

Funzioni «variadiche»

- Le funzioni che ricevono un numero variabile di parametri (come ad esempio la funzione **`printf()`**) sono dette funzioni «**variadiche**» o funzioni «varargs»:

```
tipo nome_funzione(arg1, arg2, ...);
```

- L'ellissi **...** (tre punti) indica che ci possono essere altri parametri dopo quelli fissi.
- Deve esserci almeno un parametro fisso esplicito.
- L'ellissi può essere indicata solo come ultimo elemento di una lista di parametri.
- Le macro e le definizioni da utilizzare sono contenute in **<stdarg.h>**.

Inzializzazione

- All’interno di una funzione variadica bisogna definire una variabile di tipo **va_list**, un "argument pointer" che punterà, in sequenza, a ciascuno degli argomenti aggiuntivi “anonimi”, elencati dopo quelli fissi:

```
va_list argp;
```

- La macro **va_start** inizializza la variabile **argp** in modo che punti al primo degli argomenti anonimi
- è necessario fornire il nome dell’ultimo argomento fisso (nell’esempio precedente è **arg2**):

```
va_start(argp,arg2);
```

Utilizzo

- La macro

```
va_arg(argp, type)
```

- restituisce un valore del parametro amonimo **del tipo indicato** dal parametro `type` attualmente puntato da `argp` nella lista variabile.
- Modifica `argp` in modo che punti al successivo parametro anonimo:

```
x = va_arg(argp,int);
```

- Ogni chiamata a `va_arg` preleva il successivo valore dalla lista variabile.

Terminazione

- Dopo che gli argomenti anonimi sono stati elaborati, per indicare che non si intende più scandire gli elementi di **`argp`**, si deve chiamare la macro **`va_end`** prima che termini la funzione:

```
va_end(argp);
```

- I parametri possono essere scanditi più volte: è sufficiente chiamare **`va_end`** e poi nuovamente **`va_start`**.

Esempio

```
#include <stdarg.h>
#include <stdarg.h>
void stampa(int quanti, ...);


main()
{
stampa(0);
stampa(1, "END");
stampa(2, "END", "with no exit code");
stampa(3, "END", "with code ", 8);
return 0;
}
```

Esempio

```
void stampa(int quanti, ...) {
  va_list ap;


  va_start(ap, quanti);
  if (quanti >= 1)
     printf(" %s",va_arg(ap,char*) );
  if (quanti >= 2)
    printf(" %s",va_arg(ap,char*) );
  if (quanti >= 3)
     printf(" %d",va_arg(ap,int) );
  printf("\n");
  va_end(ap);
}
```

Numero di parametri anonimi

- Per indicare alla funzione quanti sono i parametri, si può:
  - utilizzare una stringa di formato che indichi il numero e il tipo dei parametri (es. la stringa di formato di printf);
  - passare tale valore come argomento fisso;
  - terminare la lista degli argomenti facoltativi con un valore speciale (sentinella).
- Esempio
  - Chiamata:

```
maxstr(4, s0,s1,s2,s3);
```

  - Nella funzione si ha:

```
while (quanti-- > 0) {
   printf("%s", va_arg(ap, char*));
 }
```

Numero di parametri anonimi

- Per indicare alla funzione quanti sono i parametri, si può:
  - utilizzare una stringa di formato che indichi il numero e il tipo dei parametri (es. la stringa di formato di printf);
  - passare tale valore come argomento fisso;
  - terminare la lista degli argomenti facoltativi con un valore speciale (sentinella).
- Esempio
  - Chiamata:

```
maxstr(s0,s1,s2,s3,(char *)NULL);
```

  - Nella funzione si ha:

```
while ((p=va_arg(ap,char*)) != NULL) {
        printf("%s", p);
  }
```

Numero di parametri anonimi

- Per indicare alla funzione quanti sono i parametri, si può:
  - utilizzare una stringa di formato che indichi il numero e il tipo dei parametri (es. la stringa di formato di printf);
  - passare tale valore come argomento fisso;
  - terminare la lista degli argomenti facoltativi con un valore speciale (sentinella).
- Esempio
  - Chiamata:

```
sum_pos_int(1, 2, 3, 0);
```

  - Nella funzione `sum_pos_int` si ha:

```
while ((x=va_arg(ap, int)) != 0) {
        sum = sum + x;
}
```

Esempio: una semplice printf()

```
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>

#define MAX_DIGITS 30

void myprintf(const char *format, ...);

void  main()
{
   int num = 12, num2 = -95;
   char ch = 'A';

   myprintf("%d %c salve %s %d\n", num, ch, "Marco", num2);
}
```

```
OUTPUT:

> 12 A salve Marco -95
```

```
void myprintf(const char *format, ...)    {
   va_list  params;

   va_start(params, format);
   for ( ; *format != '\0' ; ++format)  {
      if (*format == '%')  {
         ++format;
         switch (*format) {
            case 's': {
                char *s = va_arg(params, char*);
                puts(s);
                break;
             }
            case 'd': {
                 char intstr[MAX_DIGITS];
                 int num = va_arg(params, int);
                 itoa(num,intstr,10);
                 puts(intstr);
                 break;
             }
            case 'c': {
                 int ch = va_arg(params, int);
                 putchar(ch);
                 break;
              }
         } // switch
      } // if
      else
         putchar(*format);
   } // for
   va_end(params);
}
```

Esercizi

1. Scrivere una funzione variadica che sia in grado di calcolare (e restituire) la somma di una sequenza arbitrariamente lunga di interi. Il prototipo della funzione deve essere:

   ```
   int summa_seq(int n_args,...)
   ```

2. Scrivere una funzione variadica che, data in ingresso una sequenza di lunghezza arbitraria di cifre intere tra 0 e 9, restituisca la stringa che corrisponde al numero che la sequenza di cifre in input rappresenta. Il prototipo della funzione deve essere:

   ```
   char *digits_to_num(int n_args,...)
   ```

3. Scrivere una funzione variadica che applichi un'arbitraria funzione *op* a parametro singolo (un elemento della sequenza) ad ogni elemento di una sequenza di (putatori a) elementi dello stesso tipo. ***Il tipo dei dati contenuto non è noto alla funzione variadica***. Il prototipo della funzione deve essere:

   ```
   void apply_op(void (op(void *)),...)
   ```