Progetto di FILTRI

Filtro che taglia i caratteri numerici (non li porta in uscita)

```
#include <stdio.h>
main()
     {         int c;
        while ((c = getchar()) != EOF)
           if ( ( c < ’0’ ) || ( c > ’9’) )
           putchar(c);
     }
```

Altri Filtri

```
#include <stdio.h>
main()
     {         int c;
        while ((c = getchar()) != EOF)
           if ( ( c != ’a’ ) && ( c != ’b’) && ( c != ’c’))
           putchar(c);
     }
```

```
 #include <stdio.h>
 main()
     {    int c;
     while ((c = getchar()) != EOF)
       if  ( c != ’\n’ )       putchar(c);
 }
```


FILTRO SIPC (SOLO_I_PRIMI_CARATTERI)

```
#include <stdio.h>
#include <string.h>

void main(argc, argv)
int argc;  char *argv[ ];
{    int c, n, count = 0;

if (argc != 2)
     { printf("errore: Usage %s numero\n", argv[0]);
       exit(1);}

c=0;
while (argv[1][c])
 { if (argv[1][c] < ’0’ || argv [1][c] > ’9’)
     { printf("Argomento numerico! NON %s\n", argv[1]);
       exit(2);}
     c++;
   }

n= atoi (argv[1]);
/* atoi da solo non basta: cosa vale atoi(1b4)?  */

   while ((c = getchar()) != EOF)
{   if  (count < n )  putchar(c);
     count++;
  }
exit (0);
}
```

FILTRO
filtro numero c_1 c_2 c_3 c_4 c_5 ... c_n

*filtra le linee che contengono almeno **numero** caratteri degli n passati come argomenti*

```
#include <stdio.h>
#include <string.h>
/* la funzione può essere definita direttamente qui */
int strchr (char *s, char c);

void main(argc, argv)
int argc;  char *argv[ ];
{    char linea[80];  int i,cont;
if (argc < 3)
  { printf("errore: pochi parametri\n"); exit(1);}
for (i = 0; (i < strlen (argv[1])) &&
                    argv[1][i] >= '0' && argv [1][i] <= '9'; i++) ;
if ( i < strlen (argv[1]) || (atoi(argv[1]) > argc-2))
  { printf("Errore sul primo argomento\n"); exit(2);}

while(gets(linea))
 { cont=0;
     for (i=2 ; i<argc ; i++) if (strchr(linea,*argv[i])) cont++;
     if (cont >= atoi(argv[1])) puts(linea);
}}

int strchr (char *s, char c)
 { while (*s)  if ( *s++ == c) return 1;
   return 0;
 }
```


FILTRO ancora

```
#include <stdio.h>
#include <string.h>
void main(argc, argv)
int argc;  char *argv[ ];
{    char linea[80];  int i,cont;

if (argc < 3)
  { puts("errore: pochi parametri"); exit(1);}

for (i = 0; (i < strlen (argv[1])) &&
                    argv[1][i] >= '0' && argv [1][i] <= '9'; i++) ;
if ( i < strlen (argv[1]) || (atoi(argv[1]) > argc-2))
  { puts("Errore sul primo argomento"); exit(2);}

while(gets(linea))
 { cont=0;
     for (i=2 ; i<argc ; i++)
          if (strstr (linea,argv[i])) cont++;
     if (cont >= atoi(argv[1])) puts(linea);
 }
}
```

strstr è definita nella libreria delle stringhe e ricerca la prima occorrenza di una stringa in un'altra

*char **strstr*(char *totalstring*, char **searchstring*)

21 Febbraio 1997

Si progetti in linguaggio C il filtro **filtro** che si possa ridirigere sia in ingresso, sia in uscita. **filtro** può accettare fino a 4 argomenti, ciascuno un carattere, di tipo diverso. I tipi accettati sono i seguenti: carattere maiuscolo, carattere minuscolo, carattere numerico, carattere non alfanumerico.

filtro deve lavorare sul file di ingresso: viene portato in uscita un insieme di caratteri pari all'insieme dei caratteri di ingresso. Per ogni carattere in ingresso, a secondo del tipo, si produce in uscita o un carattere di default o il corrispondente carattere eventualmente fornito dagli argomenti:

- per ogni carattere alfabetico maiuscolo, in uscita o 'A' (default) o dall'argomento relativo;
- per ogni carattere alfabetico minuscolo, o 'a' (default) o dall'argomento relativo;
- per ogni carattere numerico, o '0' (default) o dall'argomento relativo;
- per ogni carattere non alfanumerico, si porta in uscita o '@' (default) o dall'argomento relativo;
- i fine linea sono portati in uscita sempre.

```
#include <stdio.h>
#include <string.h>
#define MAIUSCOLO    0
#define MINUSCOLO    1
#define NUMERICO     2
#define NOTALFANUM   3
main  (argc, argv)
int argc;   char **argv;
{       int ch, j;
        char chm, chM, chn, chnot;
         int cegia [4] = {0,0,0,0};

/* controllo sul numero massimo argomenti */
  if (argc > 5)
   { printf (" errore:\n al massimo 4 argomenti"); exit (1); }
```



```
/* controllo sulla correttezza degli argomenti  e
   la loro non ripetizione*/
  for (j = 1; j < argc; j++)
  {
     if (strlen (argv[j]) > 1)
       { printf (" errore:\n al massimo 1 carattere in arg %d\n",
j);
          exit (2); }

/* il vettore cegia contiene la indicazione della presenza di
   un argomento di un tipo, le variabili ch* li memorizzano
*/
     if (argv[j][0] >= 'A' && argv[j][0] <= 'Z')
        if ( cegia [MAIUSCOLO])
        { printf (" errore:\n al massimo 1 maiuscolo\n"); exit (3); }
     else {chM = argv[j][0]; cegia [MAIUSCOLO] = 1;}

    if (argv[j][0] >= 'a' && argv[j][0] <= 'z')
         if ( cegia [MINUSCOLO])
         { printf (" errore:\n al massimo 1 minuscolo\n"); exit (3);}
    else {chn = argv[j][0]; cegia [MINUSCOLO] = 1;}

   if (argv[j][0] >= '0' && argv[j][0] <= '9')
    if ( cegia [NUMERICO])
    { printf (" errore:\n al massimo 1 maiuscolo\n"); exit (3); }
    else {chn = argv[j][0]; cegia [NUMERICO] = 1;}

   if ( ! isdigit (argv[j][0]) && ! isalpha (argv[j][0]) )
    if ( cegia [NOTALFANUM])
    { printf (" errore:\n al massimo 1 maiuscolo\n"); exit (3); }
     else {chnot = argv[j][0]; cegia [NOTALFANUM] = 1;}
  }
```

```
while ((ch = getchar ()) != EOF)
{/* non necessario, fatto a default if (ch == '\n') putchar (ch);
*/
  if (ch == '\n') putchar (ch);

 else
 {
  if (( ch >= 'A') && (ch <= 'Z'))
    if ( cegia [MAIUSCOLO]) ch = chM; else  ch = 'A';

  if (( ch >= 'a') && (ch <= 'z'))
    if ( cegia [MINUSCOLO]) ch = chm; else  ch = 'a';

  if (( ch >= '0') && (ch <= '9'))
    if ( cegia [NUMERICO]) ch = chn; else  ch = '0';

  if ( ! isdigit (ch) && ! isalpha (ch))
    if ( cegia [NOTALFANUM]) ch = chnot; else  ch = '@';

/* in ogni caso, da input o dal default
   stampa il carattere in ch */

  putchar (ch);
  }
 }
}
```


FILTRO EXPR

Scrivere in C un filtro che lavora sugli argomenti che specificano una espressione aritmetica: operatori binari **+** e **-** e valori interi. Si vuole fornire il risultato in uscita

expr 2 + 1 - 5 ⇒ ho ottenuto -2

expr 2 + - 5 + 4 ⇒ errore

```
#include <stdio.h>
#include <string.h>

void main(argc, argv)
int argc;  char *argv[ ];
{    char linea[80];
       int j, i, K, cont=0,  addsub =1;

if (argc <  2)
 { puts("Errore:  argomenti operatori e numerici"); exit(1);}
else
if ( argc % 2 == 1)
 { puts("Errore:  argomenti dispari operatori e numerici"); exit(1);}
else
{
/* controllo argomenti numerici */
  for (i = 1; i < argc; i++, i++)
     for (K = 0; (K < strlen (argv[i])); K++)
       if ( argv[i][K] < '0' || argv [i][K] > '9')
      { printf("Errore di argomento %c", argv[i][K]); exit(2);}
```

```
/* controllo argomenti operandi */
for (i = 2; i < argc - 1; i++, i++)
     if ( argv[i][0] != '+' && argv[i][0] != '-')
      {   printf("Arg: %s non operando \n", argv[i]);
          exit(2);}
}

for (i=1; i < argc; i++)
 if (i % 2 ) if (addsub) cont += atoi(argv[i]);
        else cont -= atoi(argv[i]);
 else addsub = argv[i][0] == '+' ? 1: 0;

printf(" ho ottenuto %d\n", cont);
}
```

Estendere il filtro **expr** a considerare altri operatori e confrontarlo con il comando corrispondente di UNIX