Esercitazione di Venerdì 20 Aprile 2018

1) (parametri1.c) Scrivere un programma in C che, dopo aver controllato di essere invocato con almeno 1 parametro, visualizza su standard output il valore di tutti i parametri (cioè le corrispondenti stringhe) inserendo opportune frasi che facciano capire all'utente che cosa viene visualizzato.

```
soELab@Lica02:~/file/primaEsercitazione$ cat parametri1.c
#include <stdio.h>
#include <stdlib.h>

int main  (int argc, char **argv)
{       int i; /* i serve per scorrere i parametri */

if (argc < 2)
{
        printf("Errore: Necessario almeno 1 argomento per %s\n", argv[0]);
        exit(1);
}

printf("Eseguo il programma %s con %d parametri\n", argv[0], argc -1);

for (i = 1; i < argc; i++)
        printf ("Il parametro di indice %d e' %s\n", i, argv[i]);
exit(0);
}
```

2) (parametri2.c) Scrivere un programma in C che deve essere invocato con esattamente 3 parametri: il primo deve essere considerato il nome di file, il secondo un numero intero N strettamente maggiore di 0 e il terzo deve essere considerato un singolo carattere C. Dopo aver fatto tutti i controlli necessari, si visualizza su standard output il valore di tutti i parametri (secondo il loro significato) inserendo opportune frasi che facciano capire all'utente che cosa viene visualizzato.

   ⇨ Nota Bene. Nella soluzione riportata non viene fatto nessun controllo specifico sul primo parametro dato che in questo esercizio tale parametro deve essere solo considerato come una stringa e non viene usato come nome di file.

```
soELab@Lica02:~/file/primaEsercitazione$ cat parametri2.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main  (int argc, char **argv)
{       int N; /* serve per convertire il secondo parametro */
        char C; /* serve per selezionare il terzo parametro */

if (argc != 4)
{
        printf("Errore:  Necessario  esattamente  3  argomenti  per  %s\n",
argv[0]);
        exit(1);
}

N=atoi(argv[2]);
if (N <= 0)
{
        printf("Errore: Il secondo parametro non e' un numero strettamente
maggiore di 0\n");
        exit(2);
}
```

```
if (strlen(argv[3]) != 1)
{
        printf("Errore: Il terzo parametro non e' un singolo carattere\n");
        exit(3);
}
C=argv[3][0];
printf("Eseguo il programma %s con %d parametri\n", argv[0], argc -1);
printf ("Il primo parametro e' il nome di un file %s\nIl secondo parametro e'
il numero strettamente positivo %d\nIl terzo parametro e' il singolo carattere
%c\n", argv[1], N, C);
exit(0);
}
```

3) (contaOccorrenze.c) Scrivere un programma in C che deve essere invocato con esattamente 2 parametri: il primo deve essere considerato il nome di file F, mentre il secondo deve essere considerato un singolo carattere Cx. Dopo aver fatto tutti i controlli necessari, si calcoli quante occorrenze del carattere Cx sono presenti nel file F e si visualizza tale valore su standard output inserendo opportune frasi che facciano capire all'utente che cosa viene visualizzato.

```
soELab@Lica02:~/file/primaEsercitazione$ cat contaOccorrenze.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>

int main  (int argc, char **argv)
{        long int totale=0; /* serve per calcolare il numero di occorrenze:
meglio usare un long int */
        char Cx; /* serve per selezionare il secondo parametro */
        int fd; /* per la open */
        char c; /* per leggere i caratteri dal file */

if (argc != 3)
{
        printf("Errore:  Necessario  esattamente  2  argomenti  per  %s\n",
argv[0]);
        exit(1);
}

if (strlen(argv[2]) != 1)
{
        printf("Errore: Il secondo parametro non e' un singolo carattere\n");
        exit(2);
}
Cx=argv[2][0];

printf("Eseguo il programma %s con parametri %s e %c\n", argv[0], argv[1],
Cx);
/* apriamo il file */
if ((fd = open(argv[1], O_RDONLY)) < 0)
{
        printf("Errore: FILE NON ESISTE\n"); exit(3);
}
/* leggiamo il file */
while (read (fd, &c, 1) != 0)
{         if (c == Cx) totale++;     /* se troviamo il carattere increntiamo
il conteggio */
}
printf ("Il numero totale di occorrenze del carattere %c nel file %s e' %ld\n",
Cx, argv[1], totale);
```

```
exit(0);
}
```

4) (mycat1.c) Scrivere un programma in C che, partendo dal programma mycat.c mostrato a lezione, consideri di poter essere invocato anche con un numero qualunque di nomi di file.

```
soELab@Lica02:~/file/primaEsercitazione$ cat mycat1.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char **argv)
{       char buffer [BUFSIZ];
        int nread, i, fd = 0;
        int finito=0;

printf("Eseguo il programma %s con %d parametri\n", argv[0], argc -1);

i = 1;
while (!finito)
{
        if (argc >= 2)
        /* se abbiamo dei parametri questo devono essere considerati nomi di
un file */
                if ((fd = open(argv[i], O_RDONLY)) < 0)
                {       printf("Errore in apertura file %s\n", argv[i]);
                        exit(1);
                }
        i++;
        /* se non abbiamo un parametro, allora fd rimane uguale a 0 */
        while ((nread = read(fd, buffer, BUFSIZ)) > 0 )
        /* lettura dal file o dallo standard input fino a che ci sono caratteri
*/
                write(1, buffer, nread);
                /* scrittura sullo standard output dei caratteri letti */
        if ((argc == 1) || (i == argc))  finito=1;
}

return 0;
}
```

5) (22sett99-1.c) Scrivere un programma in C che, partendo dal programma 22sett99.c mostrato a lezione, consideri un ulteriore parametro carattere che deve essere usato per sostituire il carattere cercato.

```
soELab@Lica02:~/file/primaEsercitazione$ cat 22sett99-1.c
/*
Si progetti, utilizzando il linguaggio C e le primitive di basso livello che
operano sui file, un filtro che accetta tre parametri: il primo parametro deve
essere il nome di un file (F), mentre il secondo e il terzo devono essere
considerati singoli caratteri (C1 e C2).  Il filtro deve operare una modifica
del contenuto del file F: in particolare, tutte le occorrenze del carattere
C1 nel file F devono essere sostituite con il carattere C2.
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
```

```
int main(int argc, char **argv)
{       int fd;
        char c;

if (argc != 4)
{         printf("Errore nel numero di parametri: %s vuole 3 parametri\n",
argv[0]);
        exit(1); }

if ((fd = open(argv[1], O_RDWR)) < 0)
               { printf("Errore in apertura file %s\n", argv[1]);
                 exit(2); }

if (strlen(argv[2]) != 1)
               { printf("Errore non carattere %s\n", argv[2]); exit(3); }

if (strlen(argv[3]) != 1)
               { printf("Errore non carattere %s\n", argv[3]); exit(4); }

while (read(fd, &c, 1))
          if (c == argv[2][0])
                { lseek(fd, -1L, 1);
         /* SI DEVE RIPORTARE INDIETRO IL FILE POINTER */
                  write(fd, &argv[3][0], 1);
                }
return 0;
}
```

6) (append1.c) Scrivere un programma in C che, partendo dal programma append.c mostrato a lezione, ottenga lo stesso comportamento, ma usando la open nella sua forma avanzata.

```
soELab@Lica02:~/file/primaEsercitazione$ cat append1.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#define PERM 0644               /* in UNIX */

int appendfile (char *f1)
{  int outfile, nread;  char buffer [BUFSIZ];
if ( (outfile = open (f1, O_WRONLY | O_APPEND | O_CREAT, PERM)) < 0 )
        /* apertura in append o creazione */
                        return (1);
while (( nread = read (0, buffer, BUFSIZ)) > 0 )
        /* si legge dallo standard input */
{   if ( write (outfile, buffer, nread ) < nread )
        { close (outfile); return (2); /* errore */ }
}/* fine del file di input */
close (outfile); return (0);
}

int main (int argc, char ** argv)
{ int integi;
if (argc != 2) /* controllo sul numero di argomenti */
{ printf ("ERRORE: ci vuole un argomento \n"); exit (3); }
integi = appendfile (argv[1]);
exit (integi);
}
```

7) (selezionaMultipli.c e selezionaMultipli1.c) Scrivere un programma in C che deve essere invocato con esattamente 2 parametri: il primo deve essere considerato il nome di file F, mentre il secondo un numero intero n strettamente maggiore di 0. Dopo aver fatto tutti i controlli necessari, si visualizza su standard output tutti i caratteri del file F che si trovano in posizione multipla di n.

   ⇨ Si riportano due soluzioni: selezionaMultipli.c leggere n caratteri alla volta dal file individuando sempre l'n-esimo, mentre selezionaMultipli.c utilizza la funzione lseek per andare a leggere SOLO il carattere di interesse; in entrambe le soluzioni si considera il caso che la lunghezza del file NON sia multiplo di n.

```
soELab@Lica02:~/file/primaEsercitazione$ more selezionaMultipli*.c

::::::::::::::

selezionaMultipli.c

::::::::::::::
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

int main  (int argc, char **argv)
{       int n; /* lunghezza corrispondente al secondo parametro */
        int nr; /* per la read */
        int i; /* indice del multiplo */
        char *buffer; /* per leggere i caratteri dal file */
        int fd; /* per la open */

if (argc != 3)
{
        printf("Errore: Necessario 2 argomenti per %s\n", argv[0]);
        exit (1);
}
n = atoi(argv[2]); /* convertiamo il secondo parametro */
if (n <= 0)
{
        printf("Errore: Il secondo parametro non e' un numero strettamente
maggiore di 0\n");
        exit(2);
}
fd = open(argv[1], O_RDONLY); /* apriamo il file passato come primo parametro
*/
if (fd < 0)
{
        printf("Errore: FILE NON ESISTE\n"); exit(3);
}

buffer = (char *)(malloc(n * sizeof(char)));
if (buffer == NULL)
{
        printf("Errore: problemi nella malloc\n"); exit(3);
}

printf("Stiamo per selezionare i caratteri multipli di %d\n", n);
i = 1; /* valore iniziale del multiplo */
while ((nr=read (fd, buffer, n)) > 0)
{       if (nr == n)
        {
         /* possiamo selezionare l'n-esimo carattere */
```

```
                printf("Il carattere multiplo %d-esimo all'interno del file
e' %c\n", i, buffer[n-1]);
                i++;    /* incrementiamo il conteggio */
        }
        else printf("Il file non ha una dimensione multipla di %d\n", n);
}
exit(0);
}
```

::::::::::::::

selezionaMultipli1.c

::::::::::::::

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

int main  (int argc, char **argv)
{       int n; /* lunghezza corrispondente al secondo parametro */
        long int pos; /* per la lseek */
        long int lunghezza; /* per la lunghezza del file */
        int i; /* indice del multiplo */
        char ch; /* per leggere i caratteri dal file */
        int fd; /* per la open */

if (argc != 3)
{
        printf("Errore: Necessario 2 argomenti per %s\n", argv[0]);
        exit (1);
}
n = atoi(argv[2]); /* convertiamo il secondo parametro */
if (n <= 0)
{
        printf("Errore: Il secondo parametro non e' un numero strettamente
maggiore di 0\n");
        exit(2);
}
fd = open(argv[1], O_RDONLY); /* apriamo il file passato come primo parametro
*/
if (fd < 0)
{
        printf("Errore: FILE NON ESISTE\n"); exit(3);
}

lunghezza=lseek(fd, 0L, SEEK_END);

printf("Stiamo per selezionare i caratteri multipli di %d\n", n);
i = 1; /* valore iniziale del multiplo */
pos=(long int)0; /* inizializzazione per entrare il ciclo */
while (pos < lunghezza) /* se pos minore di lunghezza */
{
        /* calcoliamo la posizione del carattere che deve essere letto */
        pos=(long int)(i * n);
        if (pos < lunghezza)
        {
                /* chiamiamo la lseek passando come offset pos-1 dall'inizio
del file: dobbiamo considerare -1 altrimenti leggeremmo il carattere sbagliato
*/
                lseek(fd, pos-1, SEEK_SET);
```

```
                read(fd, &ch, 1);
                printf("Il carattere multiplo %d-esimo all'interno del file
e' %c\n", i, ch);
                i++;    /* incrementiamo il conteggio */
        }
        else printf("Il file non ha una dimensione multipla di %d\n", n);
}
exit(0);
}
```

8) (myhead1.c, myhead2.c e myhead3.c) Scrivere un programma in C che deve essere invocato con esattamente un parametro che deve essere considerata una opzione *-numero*. Dopo aver fatto tutti i controlli necessari, il programma si deve comportare come il filtro head e quindi deve filtrare in uscita le prime *numero* linee dello standard input. Si modifichi, quindi tale programma in modo che in assenza di parametri venga considerato di filtrare le prime 10 linee, esattamente come di comporta il filtro head. Da ultimo, si modifichi ulteriormente il programma in modo che si comporti sia come filtro che come comando (filtrando le linee del file specificato invece che quelle dello standard input) e quindi potendo avere fino a due parametri (il primo deve essere considerato l'opzione –numero e il secondo il nome del file).

```
soELab@Lica02:~/file/primaEsercitazione$ more myhead*.c
::::::::::::::
myhead1.c
::::::::::::::
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main  (int argc, char **argv)
{       int i, n; /* i serve per contare le linee, n per sapere quante linee
devono essere mostrate (deriva dall'opzione) */
        char c; /* per leggere i caratteri da standard input e scriverli su
standard output */

if (argc != 2)
{
        printf("Errore: Necessario 1 argomento per %s\n", argv[0]);
        exit (1);
}
if (argv[1][0] != '-')
{
        printf ("Errore: Necessario il simbolo di opzione\n");
        exit (2);
}
else n = atoi(&(argv[1][1]));

i = 1; /* inizializzo il conteggio delle linee a 1 */
while (read (0, &c, 1) != 0)
      {        if (c == '\n') i++;     /* se troviamo un terminatore di linea
incrementiamo il conteggio */
                write(1, &c, 1);         /* scriviamo comunque il carattere
qualunque sia */
                if (i > n) break;        /* se il conteggio supera n allora
terminiamo il programma uscendo con valore di successo (0) */
      }
exit(0);
}


::::::::::::::
myhead2.c
::::::::::::::
```

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

int main  (int argc, char **argv)
{       int i, n; /* i serve per contare le linee, n per sapere quante linee
devono essere mostrate (deriva dall'opzione) */
        char c; /* per leggere i caratteri da standard input e scriverli su
standard output */

if (argc > 2)
{
        printf("Errore: Necessario 0 oppure 1 argomento per %s\n", argv[0]);
        exit (1);
}
switch (argc)
{
        case 2: /* in caso di 1 argomento questo viene interpretato come
opzione */
                if (argv[1][0] != '-')
                {
                        printf ("Errore: Necessario il simbolo di opzione\n");
                        exit (2);
                }
                else n = atoi(&(argv[1][1]));
                break;
        case 1: n = 10;
                break;
}

i = 1; /* inizializzo il conteggio delle linee a 1 */
while (read (0, &c, 1) != 0)
      {        if (c == '\n') i++;     /* se troviamo un terminatore di linea
incrementiamo il conteggio */
                write(1, &c, 1);         /* scriviamo comunque il carattere
qualunque sia */
                if (i > n) break;        /* se il conteggio supera n allora
terminiamo il programma uscendo con valore di successo (0) */
      }
exit(0);
}

::::::::::::::
myhead3.c
::::::::::::::
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

int main  (int argc, char **argv)
{       int i, n; /* i serve per contare le linee, n per sapere quante linee
devono essere mostrate (deriva dall'opzione) */
        char c; /* per leggere i caratteri da standard input o da file e
scriverli su standard output */
        int fd; /* per la open */
        int par = 0; /* per tenere traccia se e' usato un nome di file o meno
*/
        char *op, *nf; /* per rendere piu' leggibile il codice */

if (argc > 3)
```

```
{
        printf("Errore:  Necessario  0,  1  oppure  2  argomenti  per  %s\n",
argv[0]);
        exit (1);
}
switch (argc)
{
        case 3: op = argv[1]; nf = argv[2];
                par = 1;
                if (op[0] != '-')
                {
                        printf ("Errore: Necessario il simbolo di opzione\n");
                        exit (2);
                }
                else  n = atoi(&(op[1]));
                break;

        case 2: op = argv[1];
                if (op[0] != '-') { nf = op; n = 10; par = 1;}
                else  n = atoi(&(op[1]));
                break;

        case 1: n = 10;  break;
}

if (par == 1)
{       fd = open(nf, O_RDONLY);
        if (fd < 0)
        {
                printf("Errore: FILE NON ESISTE\n"); exit(3);
        }
}
else
        fd = 0; /* si deve considerare lo standard input nella lettura */

i = 1; /* inizializzo il conteggio delle linee a 1 */
while (read (fd, &c, 1) != 0)
      {        if (c == '\n') i++;     /* se troviamo un terminatore di linea
incrementiamo il conteggio */
                write(1, &c, 1);         /* scriviamo comunque il carattere
qualunque sia */
                if (i > n) break;        /* se il conteggio supera n allora
terminiamo il programma uscendo con valore di successo (0) */
      }
exit(0);
}
```

9) (selezionaLinea.c) Scrivere un programma in C che deve essere invocato con esattamente 2 parametri: il primo deve essere considerato il nome di file F, mentre il secondo un numero intero n strettamente maggiore di 0. Dopo aver fatto tutti i controlli necessari, si visualizza su standard output la linea n-esima del file F.

```
soELab@Lica02:~/file/primaEsercitazione$ cat selezionaLinea.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

int main  (int argc, char **argv)
{       int i, n; /* i serve per contare le linee, n lunghezza corrispondente
al secondo parametro */
        int j; /* indice per il buffer che contiene la linea corrente */
```

```
        char buffer[256]; /* per leggere i caratteri dal file, supponendo che
una linea non sia piùlunga di 256 comreso il terminatore di linea e quello di
stringa perche' traformeremo la linea in una stringa per poterla stampare con
printf */
        int fd; /* per la open */
        int trovata=0; /* indica se e' stata trovata la linea indicata */

if (argc != 3)
{
        printf("Errore: Necessario 2 argomenti per %s\n", argv[0]);
        exit (1);
}
n = atoi(argv[2]); /* convertiamo il secondo parametro */
if (n <= 0)
{
        printf("Errore: Il secondo parametro non e' un numero strettamente
maggiore di 0\n");
        exit(2);
}
fd = open(argv[1], O_RDONLY); /* apriamo il file passato come primo parametro
*/
if (fd < 0)
{
        printf("Errore: FILE NON ESISTE\n"); exit(3);
}

i = 1; /* inizializzo il conteggio delle linee a 1 */
j = 0; /* valore iniziale dell'indice */
while (read (fd, &(buffer[j]), 1) != 0)
{       if (buffer[j] == '\n')
        {
                if (n == i) /* trovata la linea che dobbiamo selezionare e
quindi la stampiamo dopo averla resa una stringa */
                {       buffer[j+1] = '\0';
                        printf("La linea numero %d del file %s e':\n%s", n,
argv[1], buffer);
                        trovata=1;
                        break; /* usciamo dal ciclo di lettura */
                }
                else
                {       j = 0; /* azzeriamo l'indice per la prossima linea */
                        i++;       /* se troviamo un terminatore di linea
incrementiamo il conteggio */
                }
        }
        else j++;
}
if (!trovata)
        printf("La linea numero %d non esiste in %s\n", n, argv[1]);

exit(0);
}
```

10) (selezionaLunghezzaLinea.c) Scrivere un programma in C che deve essere invocato con esattamente 2 parametri: il primo deve essere considerato il nome di file F, mentre il secondo un numero intero n strettamente maggiore di 0. Dopo aver fatto tutti i controlli necessari, si visualizza su standard output tutte le linee del file F la cui lunghezza compreso il terminatore di linea sia esattamente uguale a n.

```
soELab@Lica02:~/file/primaEsercitazione$ cat selezionaLunghezzaLinea.c
#include <stdio.h>
#include <unistd.h>
```

```
#include <stdlib.h>
#include <fcntl.h>

int main  (int argc, char **argv)
{       int i, n; /* i serve per contare le linee, n lunghezza corrispondente
al secondo parametro */
        int j; /* indice per il buffer che contiene la linea corrente */
        char buffer[256]; /* per leggere i caratteri dal file, supponendo che
una linea non sia pi▒ lunga di 256 compreso il terminatore di linea e quello
di stringa perche' trasformeremo la linea in una stringa per poterla stampare
con printf */
        int fd; /* per la open */
        int trovata=0; /* indica se e' stata trovata almeno una linea della
lunghezza indicata */
if (argc != 3)
{
        printf("Errore: Necessario 2 argomenti per %s\n", argv[0]);
        exit (1);
}
n = atoi(argv[2]); /* convertiamo il secondo parametro */
if (n <= 0)
{
        printf("Errore: Il secondo parametro non e' un numero strettamente
maggiore di 0\n");
        exit(2);
}
fd = open(argv[1], O_RDONLY); /* apriamo il file passato come primo parametro
*/
if (fd < 0)
{
        printf("Errore: FILE NON ESISTE\n"); exit(3);
}

i = 1; /* inizializzo il conteggio delle linee a 1 */
j = 0; /* valore iniziale dell'indice */
while (read (fd, &(buffer[j]), 1) != 0)
{       if (buffer[j] == '\n')
        {
                if (n == j+1) /* trovata una linea con la lunghezza giusta e
quindi la stampiamo dopo averla resa una stringa */
                {       buffer[j+1] = '\0';
                        printf("La linea numero %d del file %s ha la lunghezza
cercata cioe' %d:\n%s", i, argv[1], n, buffer);
                        trovata=1;
                }
                j = 0; /* azzeriamo l'indice per la prossima linea */
                i++;    /* se troviamo un terminatore di linea incrementiamo
il conteggio */
        }
        else j++;
}
if (!trovata)
        printf("Non e' stata trovata alcuna linea nel file %s di lunghezza
%d\n", argv[1], n);
exit(0);
}
```

Viene infine riportato un possibile makefile: in particolare, quello riportato consente di compilare e linkare tutti i file .c della directory corrente:

```
soELab@Lica02:~/file/primaEsercitazione$ more makefile
# variabile cc specifica il compilatore da utilizzare
CC=gcc
#parametro utilizzato dal compilatore C
CFLAGS=-Wall
SRC = $(wildcard *.c)
TAR = $(SRC:.c=)

all: $(TAR)

%: %.c
        $(CC) $(CFLAG)  $< -o $@
```