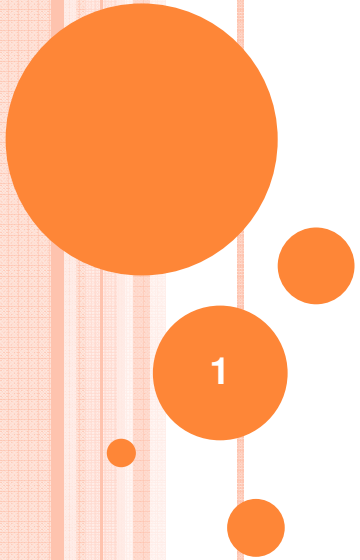
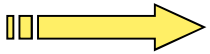


Flex (II parte)

Lunedì 14 Ottobre

I PRIMI ESEMPI...

Il programma più semplice:

primo.fl 

```
%%
```

Esiste la regola di default:
il testo che non “matcha” con nulla viene ricopiato sull’output.

Genera uno scanner che semplicemente copia il suo input (un carattere per volta) nel suo output.

N.B. Sia nella sezione delle definizioni che in quella delle regole:
- ogni testo indentato o racchiuso tra %{ e %} viene copiato parola per parola sull’output.
- %{ e %} devono apparire a inizio di linea (non indentati).
- nella sezione delle definizioni, un commento non-indentato che comincia con /* e finisce con */ diventa un commento nel file di output

Come si usa:

1. flex primo.fl
2. gcc lex.yy.c -o primo
3. primo < prova.txt >output.txt

Esempio 1

- Mostriamo come deve essere scritto il file primo.fl

```
/* E' un opzione che serve nel caso si voglia utilizzare la
funzione main generata di default */
%option main

%%
```

Alcune opzioni

%option seguito da

main: flex fornisce un routine main() di default

noyywrap: lo scanner non chiamerà la routine yywrap che serve per la gestione di più file di input

Come definire le azioni?

pattern azione

- Se l'azione è vuota (contiene solo ;), al match del pattern non succede nulla.
- Se l'azione contiene un {, allora verranno eseguite le azioni fino al raggiungimento di }.
- Un'azione che consiste solo di | indica che è definita nella stesso modo della regola che segue.

Es1:

```
[ ]    |
\t     |
\n     ;
```

Es2:

```
[a-z]+                    {printf("%s",yytext);}
```

È equivalente a

```
[a-z]+                    {ECHO;}
```

Direttive speciali che possono essere incluse in un'azione:
ECHO: copia yytext nell'output dello scanner
BEGIN: seguito dal nome di una condizione START posiziona lo scanner in tale condizione
REJECT: dirige lo scanner a procedere sul secondo longest best match (di solito un prefisso dell'input)

Esempio 2

- Costruire uno scanner che conta il numero di caratteri e il numero di linee dell'input:

```
/* Nella sezione definitions i commenti vengono copiati nel file di
   output*/
%{
    int num_lines = 0, num_chars = 0;
%}
%option noyywrap
%%
\n      {++num_lines; ++num_chars;}
.       {++num_chars;}
%%
int main(void)
        {
        yylex();
        printf( "# of lines = %d, # of chars = %d\n",
                num_lines, num_chars );
        return 0;
        }
```

Variabili e routine disponibili all'utente

- yylex() routine di scanning
- yytext stringa con la quale avviene il match
- yyin input file (default: stdin)
- yyout output file (default: stdout)
- yyleng lunghezza di yytext
- unput(c) rimette il car c nella stringa dei simboli da esaminare
- input() legge il carattere successivo
- …

Esempio 3

- Costruire uno scanner che conta il numero di caratteri e il numero di linee di un file dato in input:

```
%{
    int num_lines = 0, num_chars = 0;
%}
%option noyywrap
%%
\n      ++num_lines; ++num_chars;
.       ++num_chars;

%%
int main(int argc, char *argv[])
            {
    --argc;  /* skip over program name */
    if ( argc > 0 )
            yyin = fopen(argv[1], "r" );
    else
            yyin = stdin;

    yylex();
    printf( "# of lines = %d, # of chars = %d\n",
                num_lines, num_chars );
    return 0;
         }
```

Primi esercizi

- Scrivere un programma in Flex che :
  1. raddoppi tutte le occorrenze delle vocali in un file
  2. conti le occorrenze dei numeri multipli di 3 e dei multipli di 5
  3. comprima gli spazi e i tab in un unico singolo blank, e rimuova quelli alla fine di una linea.
  4. Elimini il testo inserito tra { e }
  5. Elimini il testo inserito tra { e } su un'unica linea
  6. Mantenga solo le linee che finiscono o cominciano con una consonante, cancellando le altre.
  7. Converta tutte le lettere maiuscole di un file in minuscole, ad eccezione di quelle racchiuse tra " e "
  8. Restituisca il più piccolo dei numeri naturali presenti nel testo
  9. Conti le occorrenze dei numeri pari e di quelli dispari

Esercizio 1

- Scrivere un programma in Flex che raddoppi tutte le occorrenze delle vocali in un file

```
%option main

%%
[aeiouAEIOU]     {ECHO;ECHO;}
```

Esercizio 1

- Scrivere un programma in Flex che raddoppi tutte le occorrenze delle vocali in un file

```
%option main

vocali    [aeiouAEIOU]
%%
{vocali}      {ECHO;ECHO;}
```

Esercizio 2

- Scrivere un programma in Flex che conti le occorrenze dei numeri multipli e dei multipli di 5

```
    int conta_3=0, conta_5=0, num;
%option noyywrap

nat 0|[1-9][0-9]*

%%
{nat}       {num=atoi(yytext);
        if (num%3==0)
            conta_3++;
        if (num%5==0)
            conta_5++;}
\n      ;
.       ;
%%
int main(void){
    yylex();
    printf("il numero dei numeri multipli di 3 è:
%d",conta_3);
    printf("il numero dei numeri multipli di 5 è:
%d",conta_5);
    return 0;
}
```

Esercizio

- Scrivere un programma in Flex che converta un file attraverso un cifrario a sostituzione.
  Esso consiste nel triplicare ogni vocale con l'aggiunta di una g e una s tra le occorrenze: per esempio, a diventa agasa, e diventa egese, e così via (quindi "ciao" diventa cigisiagasaogoso).
  Assumiamo che il file sia una sequenza di parole separati da spazi.