P1
werkcollege 4

patroon vervangen
structuren
finite state machines

opdracht 3

- maak 'n functie
  vervangAlle(char string [], char oud [], char nieuw [])
- string wordt veranderd
- bijvoorbeeld
  char voorbeeld [10] = "C++";
  vervangAlle(voorbeeld,"+","--");
  cout << voorbeeld << endl;
- geeft "C----"

aanpak

- splits in kleinere onderdelen

while (nietAanEinde)
  vervang (string, oud, nieuw, pos)

- voeg argumenten tot voor communicatie tussen functies

in C++

```
void replaceAll( char string [], char oud [],
                                 char nieuw [])
{
  if ( strlen ( oud ) > 0 )
      for ( int pos=0;
            (pos=past(string,oud,pos))>=0;
            pos=pos+strlen(nieuw)-strlen(oud)+1)
      vervang ( string, oud, nieuw, pos);
}
```

aanpak 2

- vervang (string, oud, nieuw, pos)
- ook splitsen in onderdelen
  - zoek patroon
  - pas ruimte aan
  - kopieer nieuw naar de juiste positie
- voeg weer argumenten en functieresultaten toe voor communicatie tussen functies
  - b.v. postie waarop patroon gevonden is
  - -1 betekent niet gevonden

5

vervangen in C++

```
void vervang (char string [], char oud [],
              char nieuw [], int pos)
{
  schuif ( string, pos, strlen( oud ),
                        strlen( nieuw ));
  kopieer( string, nieuw, pos );
}
```

6

schuiven

```
void schuif(char string [], int pos, int oud, int nieuw)
{
   const int strLen = strlen ( string );
   const int delta = nieuw-oud;
   if ( delta<0 )
        for ( int i = pos+nieuw; i<=strLen; i=i+1)
              string [ i ] = string [ i-delta ];
   else if ( delta>0 )
        for ( int i = strLen+1; i>=pos+oud; i=i-1)
              string [ i+delta ] = string [ i ];
   // else doe niets
}
```

7

kopiëren

```
void kopieer ( char string [],
               char nieuw [], int pos)
{
  for ( int i=0;nieuw[i]!='\0';i=i+1 )
      string [ pos+i ] = nieuw [ i ];
}
```

8

passen

```
int past ( char string [], char pat [],
                              int start=0)
{
  for ( int p=start; string [p] != '\0';
                                    p=p+1)
      if ( pastHier ( string, pat, p ))
            return p;
  return -1;
}
```

9

passen 2

```
bool pastHier ( char string [],
                  char pat [], int pos)
{
  int i=0;
  for ( ; string[pos+i]==pat[i] &&
          pat[i] != '\0'; i=i+1 )
     ; // lege loop body
  return pat[i]=='\0';
}
```

10

voorwaarden & randgevallen

- mag string leeg zijn?
  - en oud en nieuw?
- nieuw langer dan oud
  - nieuw korter dan oud
- patroon aan begin
- patroon aan eind
- patroon komt niet voor
- patroon komt vaak voor

11

opdracht 4

finite state machines

12

structuur

- plakt aantal elementen samen
  - ieder element heeft een naam
  - ieder element heeft eigen type

```
struct Paar
{
  int     getal;
  char    letter;
};
```

let op:
ook deze puntkomma
moet er staan !

aanmaken van structuur

- geef beginwaarden tussen { en }
  - zoals bij een lijst

```
Paar eerste = { 0, 'a' };
Paar paar = { 42, 'Z' };
Paar nogEen = { 7, '7' };

Paar rij [] = {{ 0, 'a' },{ 26, 'Z' }};
```


selectie via de namen

- selectie geeft een waarde van dat type

```
cout << paar.getal*6
     << paar.letter << endl ;
```

- dit geeft

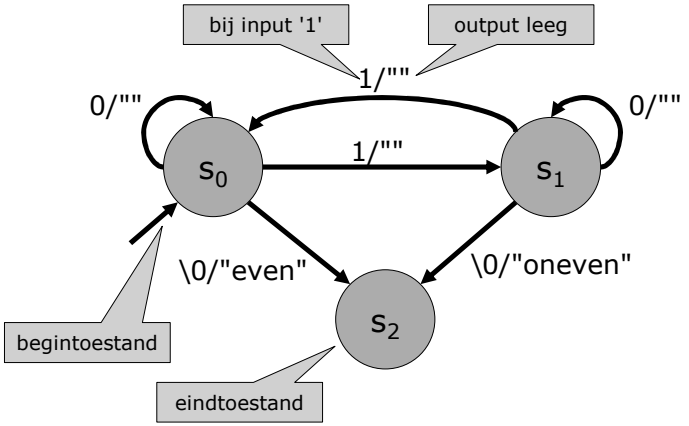
```
42Z
```


veranderen van velden

- als bij selectie

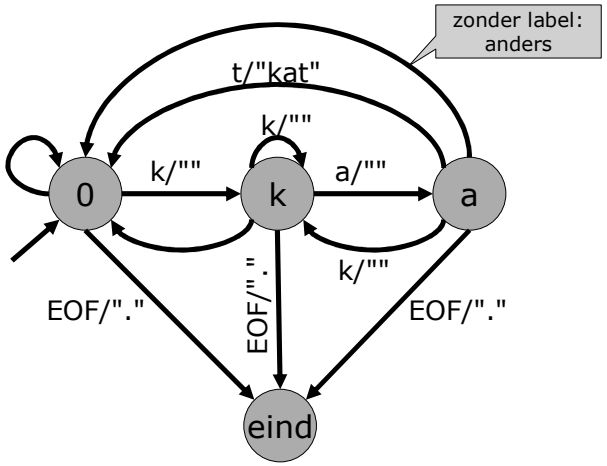
```
eerste.getal = -1;
eerste.getal = paar.getal * 3;
```

finite state machine



17

herken katten



18

in deze opgave

- invoer: char
- state: int
  - -1 is eindtoestand
  - andere toestanden opvolgende nummers vanaf 0
- uitvoer: c-string

19

overgangen

- per toestand een rijtje van overgangen
- maak hier een rijtje van
  - handig: state naam is index in deze rij

```
const int OutputLengte = 30;
typedef int State;
typedef char Input;
typedef char Output [ OutputLengte ];
```

20

een enkele overgang in C++

```
struct Trans
{
  Input in;
  State next;
  Output out;
};
```

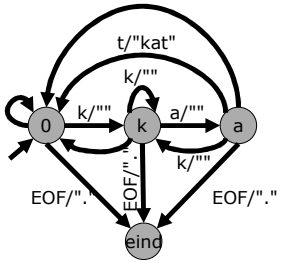
21

tabel van overgangen

```
const int Nstates = 20;
const int MaxTrans = 10;
typedef Trans Table [Nstates][MaxTrans];
```

22

tabel voor katten




```
Table tableKat
    = {{{'k',1,""},{'\0',EindState,".\n"},{AnyChar,0,""}}
     ,{{'k',1,""},{'a',2,""}, {'\0',EindState,".\n"},
        {AnyChar,0,""}}
     ,{{'k',1,""}, {'t',0,"kat"}, {'\0',EindState,".\n"},
        {AnyChar,0,""}}
     };
```

23

wanneer is een tabel gezond?

- alleen geldige toestanden komen voor
- voor iedere toestand een overgang voor *anyChar*
- niet meer overgangen voor zelfde invoer
- niet gebrukte overgangen mogen alles zijn
  - kijk daar niet naar !

24

belangrijkste functies voor FSM

- delta: zoek nieuwe toestand

State delta(Table table, State s, Input i)

- sigma: druk uitvoer af

void sigma(Table table, State s, Input i)

- 'gewoon' opzoeken in tabel

25

maak zelf een fsm

zet simpele tekst om in HTML

regel een

regel twee
o punt 1
o punt
2

laatste regel.

regel een
**
**
regel twee
<ul>
<li>
punt 1
**
**
punt
2
**
**
**
**
laatste regel.

26