Implementace KA

viz slajdy, kde jsou

3 způsoby (A) implementujeme stav místem programu a tabulku přechodů implementujeme programovou strukturou. = (S) funguje jako návěští

(2) stav se implementuje proměnnou a tabulka přechodů strukturou

(3) stav se implementuje proměnnou, ale tabulka přechodů se použije jako pole

(PŘ) implementace (2)

```
char ~~finder~~ nazev [...];
int lexan();
double hodnota;
{ int stav = S;          // počáteční
  for(;;) {
    relop = eli_relop();
    switch(stav) {
      case S: switch(relop) {
                case '/': stav=A; break;
                case '␣': ~~sto~~ break;
```

```
            case '*' : stav = E ; break ;
            :
            case '0' :
                 '1' :
                 :
                 '9' : stav = J ;
                       A3(
                       A3() ;   //akce A3
                       break ;
            default : if (vstup == EOF)
                                      ↳ = ε
                       return T_konec ;
                       else CHYBA() ;
            }
     break ;
     case E : switch (vstup) {    //konecne
            case '=' : stav = F ; break ;
            default : vrat_vstup (vstup)
                      //bude znovu
                      ctena pristim
                      cti_vstup
                   return T_brak ;
            }
            break ;
```

```
case L: if (vstup >= '0' && vstup <= '9'){
        stav = K; A7(); break; }
        else CHYBA();
```

//nekoncové

Reprezentace stavu místem v programu I

- Přechodový diagram budeme chápat jako speciální formu vývojového diagramu
- Stavům automatu budou odpovídat místa v programu označená návěštími.
- Není-li uzel *q* koncový a vedou-li z něj hrany ohodnocené symboly *a*1, *a*2, ...,resp. *an* do uzlů *p*1, *p*2, ... resp. *pn* , pak příkaz označený návěštím *q* bude mít tvar:

```
q: switch (Vstup) {
      case a1: CtiVstup(); goto p1;
      case a2: CtiVstup(); goto p2;
      ...
      case an: CtiVstup(); goto pn;
      default: return 0;
   }
```


Reprezentace stavu místem v programu I

- Je-li uzel *q* koncový a vedou-li z něj hrany ohodnocené symboly *a*1, *a*2, ..., resp. *an* do uzlů *p*1, *p*2, ..., resp. *pn*, pak příkaz označený návěštím *q* bude mít tvar:

```
q: switch (Vstup) {
      case a1: CtiVstup(); goto p1;
      case a2: CtiVstup(); goto p2;
      ...
      case an: CtiVstup(); goto pn;
      case Konec: return 1;
      default: return 0;
   }
```

Reprezentace stavu místem v programu I

- Příklad: identifikátory a celá čísla

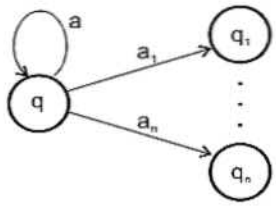
```
typedef enum {Pismeno,Cislice,Jiny,Konec} Vstupy;
Vstupy vstup;
void CtiVstup(void)
/* stejná definice jako v příkl. 2.13 */
{ ... }

int Automat(void)
{
   CtiVstup();
Start:
   switch (Vstup) {
    case Pismeno:
        CtiVstup(); goto Ident;
    case Cislice:
        CtiVstup(); goto Cislo;
    default:
        return 0;
   }
```


Reprezentace stavu místem v programu I

```
Ident:
   switch (Vstup) {
    case Pismeno:
    case Cislice:
      CtiVstup(); goto Ident;
    case Konec:
       return 1;
    default:
      return 0;
   }
Cislo:
   switch (Vstup) {
    case Cislice:
      CtiVstup(); goto Cislo;
    case Konec:
      return 1;
    default:
      return 0;
   }
}
```

Reprezentace stavu místem v programu II




```
/*q*/ while (Vstup == a) CtiVstup();
      switch (Vstup) {
       case a1: CtiVstup(); /*q1*/ Q1
       ...
       case an: CtiVstup(); /*qn*/ Qn
       case Konec: return 1; _/* jen je-li q koncový stav
       default: return 0;
      }
```

Reprezentace stavu místem v programu II

- Příklad: identifikátory a čísla

```
int Automat(void)
{
   CtiVstup();
/*Start*/
   switch (Vstup) {
    case Pismeno:
        CtiVstup();
        /*Ident*/
        while (Vstup==Pismeno || Vstup==Cislice)
           CtiVstup();
        switch (Vstup) {
         case Konec: return 1;
         default: return 0;
        }
```

Reprezentace stavu místem v programu II

```
    case Cislice:
        CtiVstup();
        /*Cislo*/
        while (Vstup==Cislice) CtiVstup();
        switch (Vstup) {
         case Konec: return 1;
         default: return 0;
        }
    default:
        return 0;
   }
}
```