Příklady pro cvičení 5. z IFJ:
Syntaktická analýza shora dolů

Příklad 1.

Uvažujte jednoduchý programovací jazyk, jehož syntaxi popisuje následující bezkontextová gramatika:

G = (*N*, *T*, *P*, **<prog>**), kde:

- *N* = {**<prog>**, **<st-list>**, **<stat>**, **<it-list>**, **<item>**},
- *T* = {**begin**, **end**, **;**, **read**, **write**, **add**, **:=**, **int**, **id**},
- *P* = {
  - 1: **<prog>** → **begin** **<st-list>** **end**
  - 2: **<st-list>** → **<stat>** **;** **<st-list>**
  - 3: **<st-list>** → ε
  - 4: **<stat>** → **read** **id**
  - 5: **<stat>** → **write** **<item>**
  - 6: **<stat>** → **id** **:=** **add** **<item>** **<it-list>**
  - 7: **<stat>** → ε *// prázdný příkaz*
  - 8: **<it-list>** → **<item>** **<it-list>**
  - 9: **<it-list>** → ε
  - 10: **<item>** → **int**
  - 11: **<item>** → **id** }

a) Sestrojte LL-tabulku pro danou gramatiku

b) Proveďte pomocí prediktivního syntaktického analyzátoru řízeného tabulkou z bodu a) syntaktickou analýzu vstupního řetězce **begin write int ; end**

Řešení:

a) Konstrukci LL-tabulky provedeme následujícím způsobem:

1) Nejprve vypočítáme množiny *First*(*X*) a *Empty*(*X*) pro každé $X \in N \cup T$ pomocí algoritmu:

I) Pro každý terminální symbol $\boldsymbol{a} \in T$ nejprve polož:
- $First(\boldsymbol{a}) = \{\boldsymbol{a}\}$; $Empty(\boldsymbol{a}) = \varnothing$

II) Pro každý neterminální symbol $\boldsymbol{A} \in N$ nejprve polož:
- $First(\boldsymbol{A}) = \varnothing$
- Pokud existuje pravidlo $\boldsymbol{A} \rightarrow \varepsilon \in P$ potom $Empty(\boldsymbol{A}) = \{\varepsilon\}$ jinak $Empty(\boldsymbol{A}) = \varnothing$

III) Procházej postupně „ve vhodném pořadí“ jednotlivá pravidla (ne tvaru $\boldsymbol{A} \rightarrow \varepsilon$) a dělej: nechť aktuální pravidlo má tvar $\boldsymbol{A} \rightarrow \boldsymbol{X_1X_2 \dots X_{n-1}X_n}$, potom:
- Přidej všechny prvky z množiny $First(X_1)$ do množiny $First(A)$
- Pokud $Empty(X_1) = \{\varepsilon\}$, potom:
  přidej všechny prvky z množiny $First(X_2)$ do množiny $First(A)$
- Pokud $Empty(X_1) = Empty(X_2) = \{\varepsilon\}$, potom:
  přidej všechny prvky z množiny $First(X_3)$ do množiny $First(A)$
  …
- Pokud $Empty(X_1) = Empty(X_2) = \dots = Empty(X_{n-1}) = \{\varepsilon\}$, potom:
  přidej všechny prvky z množiny $First(X_n)$ do množiny $First(A)$
- Pokud $Empty(X_1) = Empty(X_2) = \dots = Empty(X_{n-1}) = Empty(X_n) = \{\varepsilon\}$, potom:
  polož **!!!** $Empty(A) = \{\varepsilon\}$ **!!!**

IV) Pokud byla některá z množin změněna, znovu proveď krok III. pro všechna pravidla

Poznámky k algoritmu:

- Je vidět, že pro každý terminální symbol $\boldsymbol{a} \in T$ platí i na závěr celého algoritmu *First*($\boldsymbol{a}$) = {$\boldsymbol{a}$}; *Empty*($\boldsymbol{a}$) = $\varnothing$. Proto tyto množiny většinou nevypisujeme, ale pro výpočet tento fakt musíme znát. Stačí se tedy omezit na výpočet množin *First*($\boldsymbol{A}$), *Empty*($\boldsymbol{A}$) pro každý neterminální symbol $\boldsymbol{A} \in N$.
- Doporučuji nejdříve kompletně určit množiny *Empty*($\boldsymbol{A}$) pro každý neterminální symbol $\boldsymbol{A} \in N$, potom až určit množiny *First*($\boldsymbol{A}$) pro každý neterminální symbol $\boldsymbol{A}$. Podstatně to urychlí výpočet těchto množin.
- Doporučuji pravidla seřadit do posloupnosti tak, aby platilo: Pokud *i*-té pravidlo má na levé straně neterminál $\boldsymbol{A}$, pak každé následující pravidlo neobsahuje nikde na pravé straně neterminál $\boldsymbol{A}$. Ne vždy to jde! Pokud to jde, pak *vhodné pořadí* pro krok III) je procházení této posloupnosti od posledního pravidla k prvnímu. Podstatně to urychlí výpočet množin *First*($\boldsymbol{A}$). Pravidla gramatiky v tomto příkladu jsou tak seřazena.

Výpočet množin *First*(*X*) a *Empty*(*X*) pro výše uvedený příklad:

- Po provedení kroků I) a II) jsou tyto množiny následující:

First(**begin**)	= {**begin**}	*First*(**add**)	= {**add**}	*Empty*(**<prog>**)	= $\varnothing$
First(**end**)	= {**end**}	*First*(**:=**)	= {**:=**}	*Empty*(**<st-list>**)	= {ε}
First(**;**)	= {**;**}	*First*(**int**)	= {**int**}	*Empty*(**<stat>**)	= {ε}
First(**read**)	= {**read**}	*First*(**id**)	= {**id**}	*Empty*(**<it-list>**)	= {ε}
First(**write**)	= {**write**}			*Empty*(**<item>**)	= $\varnothing$

(Vypsány jsou pouze množiny *First* pro terminální symboly a množiny *Empty* pro neterminální symboly, neboť všechny ostatní množiny jsou po provedení kroků I) a II) vždy prázdné!)

- Dále provedeme krok III):
  - Nejprve dopočítáme kompletně množiny *Empty*($\boldsymbol{A}$) pro všechna $\boldsymbol{A} \in N$. Je vidět, že neexistuje již žádné pravidlo tvaru $\boldsymbol{A} \rightarrow X_1X_2\ldots X_{n-1}X_n$, pro které by platilo $Empty(X_1) = Empty(X_2) = \ldots = Empty(X_{n-1}) = Empty(X_n) = \{\varepsilon\}$. Množiny *Empty*($\boldsymbol{A}$) pro všechna $\boldsymbol{A} \in N$ jsou již ve finálním tvaru.
  - Nyní dopočítáme kompletně množiny *First*($\boldsymbol{A}$) pro všechna $\boldsymbol{A} \in N$. Budeme tedy procházet postupně všechna pravidla gramatiky (ne ε-pravidla) od posledního k prvnímu (viz. třetí poznámka k algoritmu):

11: **<item>** → **id** {Přidáme *First*(**id**) do *First*(**<item>**)}

10: **<item>** → **int** {Přidáme *First*(**int**) do *First*(**<item>**)}

Celkové změny: *First*(**<item>**) = {**id**, **int**}

8: **<it-list>** → **<item>** **<it-list>** {Přidáme *First*(**<item>**) do *First*(**<it-list>**)}
Empty(**<item>**) ≠ {ε}, nic jiného tedy přidávat nebudeme

Celkové změny: *First*(**<it-list>**) = {**id**, **int**}

6: **<stat>** → **id** **:=** **add** **<item>** **<it-list>** {Přidáme *First*(**id**) do *First*(**<stat>**)}
Empty(**id**) ≠ {ε}, nic jiného tedy přidávat nebudeme

5: **<stat>** → **write** **<item>** {Přidáme *First*(**write**) do *First*(**<stat>**)}
Empty(**write**) ≠ {ε}, nic jiného tedy přidávat nebudeme

4: <stat> → **read id** {Přidáme *First*(**read**) = do *First*(**<stat>**)}
Empty(**read**) ≠ {ε}, nic jiného tedy přidávat nebudeme

Celkové změny: *First*(**<stat>**) = {**id**, **write**, **read**}

2: <st-list> → **<stat> ; <st-list>** {Přidáme *First*(**<stat>**) do *First*(**<st-list>**)}
!!! *Empty*(**<stat>**) = {ε}, budeme přidávat dále:
{Přidáme *First*(**;**) do *First*(**<st-list>**)}
Empty(**;**) ≠ {ε}, nic jiného tedy přidávat nebudeme

Celkové změny: *First*(**<st-list>**) = {**id**, **write**, **read**, **;**}

1: <prog> → **begin <st-list> end** {Přidáme *First*(**begin**) do *First*(**<prog>**)}
Empty(**begin**) ≠ {ε}, nic jiného tedy přidávat nebudeme

Celkové změny: *First*(**<prog>**) = {**begin**}

Jednotlivé množiny pro neterminální symboly po provedení jednoho cyklu ve III. kroku algoritmu:

First(**<prog>**)	= {**begin**}	*Empty*(**<prog>**)	= ∅
First(**<st-list>**)	= {**id**, **write**, **read**, **;**}	*Empty*(**<st-list>**)	= {ε}
First(**<stat>**)	= {**id**, **write**, **read**}	*Empty*(**<stat>**)	= {ε}
First(**<it-list>**)	= {**id**, **int**}	*Empty*(**<it-list>**)	= {ε}
First(**<item>**)	= {**id**, **int**}	*Empty*(**<item>**)	= ∅

Pokud bychom provedli další cyklus III. kroku algoritmu, zjistili bychom, že žádná jiná množina již změněna nebyla. Hodnoty jednotlivých množin jsou tedy výsledné. Pokud bychom však pravidla procházeli v jiném pořadí, nemuseli bychom být hotovi pouze jedním průchodem!

2) Pomocí množin *First*(X) a *Empty*(X) pro každé $X \in N \cup T$ můžeme snadno určit pro daný řetězec $x \in (N \cup T)^+$ množiny *First*(x) a *Empty*(x) následujícími algoritmy:

nechť $x = X_1X_2\ldots X_{n-1}X_n$, potom *First*($X_1X_2\ldots X_{n-1}X_n$) určíme:

- *First*($X_1X_2\ldots X_{n-1}X_n$) polož ∅
- Přidej všechny prvky z množiny *First*(X_1) do množiny *First*($X_1X_2\ldots X_{n-1}X_n$)
- Pokud *Empty*(X_1) = {ε}, potom:
  přidej všechny prvky z množiny *First*(X_2) do množiny *First*($X_1X_2\ldots X_{n-1}X_n$)
- Pokud *Empty*(X_1) = *Empty*(X_2) = {ε}, potom:
  přidej všechny prvky z množiny *First*(X_3) do množiny *First*($X_1X_2\ldots X_{n-1}X_n$)
  …
- Pokud *Empty*(X_1) = *Empty*(X_2) = … = *Empty*(X_{n-1}) = {ε}, potom:
  přidej všechny prvky z množiny *First*(X_n) do množiny *First*($X_1X_2\ldots X_{n-1}X_n$)

nechť $x = X_1X_2\ldots X_{n-1}X_n$, potom *Empty*($X_1X_2\ldots X_{n-1}X_n$) určíme:

- Pokud *Empty*(X_1) = *Empty*(X_2) = … = *Empty*(X_{n-1}) = *Empty*(X_n) = {ε}, potom:
  Empty($X_1X_2\ldots X_{n-1}X_n$) = {ε} jinak *Empty*($X_1X_2\ldots X_{n-1}X_n$) = ∅

Poznámka: Pro prázdný řetězec ε dodefinujeme: *Empty*(ε) = {ε}; *First*(ε) = ∅

***Ilustrační příklad*:**
Určeme množiny *First*(**<stat> <st-list> <it-list>**) a *Empty*(**<stat> <st-list> <it-list>**).

- *First*(**<stat> <st-list> <it-list>**) = ∅

<stat> <st-list> <it-list>

- Přidáme *First*(**<stat>**) = {**id**, **write**, **read**} do *First*(**<stat> <st-list> <it-list>**)
  Celkové změny: *First*(**<stat> <st-list> <it-list>**) = {**id**, **write**, **read**}

<stat> <st-list> <it-list>

- Protože *Empty*(**<stat>**) = {ε}, přidáme rovněž *First*(**<st-list>**) = {**id**, **write**, **read**, **;**}
  do *First*(**<stat> <st-list> <it-list>**), tedy
  Celkové změny: *First*(**<stat> <st-list> <it-list>**) = {**id**, **write**, **read**, **;**}

<stat> <st-list> <it-list>

- Protože *Empty*(**<stat>**) = *Empty*(**<st-list>**) = {ε}, přidáme rovněž *First*(**<it-list>**) = {**id**, **int**}
- do *First*(**<stat> <st-list> <it-list>**), tedy
  Celkové změny: *First*(**<stat> <st-list> <it-list>**) = {**id**, **write**, **read**, **;**, **int**}

  Výsledek: *First*(**<stat> <st-list> <it-list>**) = {**id**, **write**, **read**, **;**, **int**}

- Protože *Empty*(**<stat>**) = *Empty*(**<st-list>**) = *Empty*(**<it-list>**), platí:
  Empty(**<stat> <st-list> <it-list>**) = {ε}

Poznámka: Na tomto příkladě je pouze ilustrován výpočet množin *First*(x) a *Empty*(x) pro konkrétní řetězec x = **<stat> <st-list> <it-list>**, což bude potřeba určovat pro různé řetězce při výpočtu množin *Follow* a *Predict* (viz. dále). Tento řetězec byl zvolen zcela náhodně pro vhodnou ilustraci výpočtu těchto množin, nemá tedy žádný konkrétní význam pro konstrukci *LL*-tabulky.

3) Nyní již můžeme vypočítat množinu *Follow*($\boldsymbol{A}$) pro každé $\boldsymbol{A} \in N$ pomocí algoritmu:
   I) Pro startující neterminální symbol $\boldsymbol{S}$ položíme *Follow*($\boldsymbol{S}$) = {**$**},
   pro ostatní neterminály $\boldsymbol{A} \in N$ položíme *Follow*($\boldsymbol{A}$) = ∅.

   II) Procházej postupně „ve vhodném pořadí“ jednotlivá pravidla (obsahující na pravé straně aspoň jeden neterminální symbol) a pro každé z nich udělej všechny možné dekompozice na tvar: $\boldsymbol{A} \rightarrow \boldsymbol{xBy}$, kde $\boldsymbol{x}, \boldsymbol{y} \in (N \cup T)^*$, $\boldsymbol{B} \in N$:
   - Pokud $\boldsymbol{y} \neq \varepsilon$, potom:
     přidej všechny prvky z množiny *First*($\boldsymbol{y}$) do množiny *Follow*($\boldsymbol{B}$)
   - Pokud *Empty*($\boldsymbol{y}$) = {ε} (tj. zahrnuje i možnost $\boldsymbol{y} = \varepsilon$), potom:
     přidej všechny prvky z množiny *Follow*($\boldsymbol{A}$) do množiny *Follow*($\boldsymbol{B}$)

   III) Pokud byla některá z množin změněna, znovu proveď krok II. pro všechna pravidla

Poznámky k algoritmu:

- Všimněte si, že množiny *First*(*y*) a *Empty*(*y*) musejí být určeny obecně pro **!řetězec!** *y* (ne pouze symbol).
- Pokud máme pravidla seřazena tak, jak bylo doporučeno v poznámce pro algoritmus výpočtu množin *First*(*X*) a *Empty*(*X*) pro každé $X \in N \cup T$, pak *vhodné pořadí* pro krok II. je procházení této posloupnosti pravidel od prvního k poslednímu.

Výpočet množin *Follow*(*A*) pro výše uvedený příklad:

- Po provedení kroku I) jsou tyto množiny následující:

Follow(**<prog>**)	= {**$**}
Follow(**<st-list>**)	= ∅
Follow(**<stat>**)	= ∅
Follow(**<it-list>**)	= ∅
Follow(**<item>**)	= ∅

- Nyní dopočítáme kompletně množiny *Follow*(***A***) pro všechna $A \in N$. Budeme tedy procházet postupně všechna pravidla gramatiky (ne pravidla, které neobsahují žádný neterminální symbol) od prvního k poslednímu (viz. druhá poznámka k algoritmu):

1: <prog> → begin <st-list> end

Veškeré dekompozice toho pravidla:

- <prog> → **begin** **<st-list>** **end**

  end ≠ ε

  {Přidáme *First*(**end**) = {**end**}do *Follow*(**<st-list>**)}

Celkové změny: *Follow*(**<st-list>**) = {**end**}

2: <st-list> → <stat> ; <st-list>

Veškeré dekompozice toho pravidla:

- <st-list> → **<stat> ;** **<st-list>**

  Empty(ε) = {ε}

  {Přidáme *Follow*(<st-list>) do *Follow*(**<st-list>**) // *nemá žádný význam* }

- <st-list> → **<stat>** **; <st-list>**

  ≠ ε

  {Přidáme *First*(**; <st-list>**) = {**;**} do *Follow*(**<stat>**) }

Celkové změny: *Follow*(**<stat>**) = {**;**}

5: <stat> → write <item>

Veškeré dekompozice toho pravidla:

- <stat> → **write** **<item>**

  Empty(ε) = {ε}

  {Přidáme *Follow*(<stat>) = {**;**} do *Follow*(**<item>**) }

Celkové změny: *Follow*(**<item>**) = {**;**}

6: **<stat>** → **id := add <item> <it-list>**

Veškeré dekompozice toho pravidla:

- <stat> → **id := add <item> <it-list>**

  Empty(ε) = {ε}

  {Přidáme *Follow*(<stat>) = {**;**} do *Follow*(**<it-list>**) }
- <stat> → **id := add <item> <it-list>**

  ≠ ε

  {Přidáme *First*(**<it-list>**) = {**id**, **int**} do *Follow*(**<item>**) }
- <stat> → **id := add <item> <it-list>**

  Empty(**<it-list>**) = {ε}

  {Přidáme *Follow*(<stat>) = {**;**} do *Follow*(**<item>**) // *ten tam již je* }

Celkové změny: *Follow*(**<it-list>**) = {**;**}, *Follow*(**<item>**) = {**id**, **int**, **;** }

8: **<it-list>** → **<item> <it-list>**

Veškeré dekompozice toho pravidla:

- <it-list> → **<item> <it-list>**

  Empty(ε) = {ε}

  {Přidáme *Follow*(<it-list>) do *Follow*(**<it-list>**) // *nemá žádný význam* }
- <it-list> → **<item> <it-list>**

  ≠ ε

  {Přidáme *First*(**<it-list>**) = {**id**, **int**} do *Follow*(**<item>**) // *ty tam již jsou* }
- <it-list> → **<item> <it-list>**

  Empty(**<it-list>**) = {ε}

  {Přidáme *Follow*(<it-list>) = {**;**} do *Follow*(**<item>**) // *ten tam již je* }

Jednotlivé množiny pro neterminální symboly po provedení jednoho cyklu ve II. kroku algoritmu:

Follow(**<prog>**)	= {**$**}
Follow(**<st-list>**)	= {**end**}
Follow(**<stat>**)	= { **;** }
Follow(**<it-list>**)	= { **;** }
Follow(**<item>**)	= {**id**, **int**, **;** }

Pokud bychom provedli další cyklus II. kroku algoritmu, zjistili bychom, že žádná jiná množina již změněna nebyla. Hodnoty jednotlivých množin jsou tedy výsledné.

4) Jako poslední můžeme spočítat množinu *Predict*(*r*) pro každé !**pravidlo**! ***r***: ***A*** → *x*. (ne pro symboly a řetězce!) podle následující algoritmu:
   - Pro každé pravidlo tvaru ***r***: ***A*** → *x* urči množinu *Predict*(*r*) jako:

     Pokud *Empty*(*x*) = {ε} polož *Predict*(*r*) = *First*(*x*) ∪ *Follow*(***A***);

     jinak polož *Predict*(*r*) = *First*(*x*).

Poznámka k algoritmu:

- Všimněte si, že množina *First*(*x*) musí být obecně určena pro !**řetězec**! *x* (ne pouze symbol).

Výpočet množin *Predict*(*r*) pro výše uvedený příklad:

1: <prog> → **begin** **<st-list>** **end**
Empty(**begin** **<st-list>** **end**) ≠ {ε}, tedy *Predict*(1) = *First*(**begin** **<st-list>** **end**) = {**begin**}

2: <st-list> → **<stat>** **;** **<st-list>**
Empty(**<stat>** **;** **<st-list>**) ≠ {ε}, tedy *Predict*(2) = *First*(**<stat>** **;** **<st-list>**) = {**id**, **write**, **read**, **;**}

3: <st-list> → ε
Empty(ε) = {ε}, tedy *Predict*(3) = *First*(ε) ∪ *Follow*(**<st-list>**) = ∅ ∪ {**end**} = {**end**}

4: <stat> → **read** **id**
Empty(**read** **id**) ≠ {ε}, tedy *Predict*(4) = *First*(**read** **id**) = {**read**}

5: <stat> → **write** **<item>**
Empty(**write** **<item>**) ≠ {ε}, tedy *Predict*(5) = *First*(**write** **<item>**) = {**write**}

6: <stat> → **id** **:=** **add** **<item>** **<it-list>**
Empty(**id** **:=** **add** **<item>** **<it-list>**) ≠ {ε}, tedy *Predict*(6) = *First*(**id** **:=** **add** **<item>** **<it-list>**) = {**id**}

7: <stat> → ε
Empty(ε) = {ε}, tedy *Predict*(7) = *First*(ε) ∪ *Follow*(**<stat>**) = ∅ ∪ { **;** } = { **;** }

8: <it-list> → **<item>** **<it-list>**
Empty(**<item>** **<it-list>**) ≠ {ε}, tedy *Predict*(8) = *First*(**<item>** **<it-list>**) = {**id**, **int**}

9: <it-list> → ε
Empty(ε) = {ε}, tedy *Predict*(9) = *First*(ε) ∪ *Follow*(**<it-list>**) = ∅ ∪ { **;** } = { **;** }

10: <item> → **int**
Empty(**int**) ≠ {ε}, tedy *Predict*(10) = *First*(**int**) = {**int**}

11: <item> → **id**
Empty(**id**) ≠ {ε}, tedy *Predict*(11) = *First*(**id**) = {**id**}

Jednotlivé množiny *Predict* pro všechna pravidla gramatiky:

Predict(1)	= {**begin**}
Predict(2)	= {**id**, **write**, **read**, **;**}
Predict(3)	= {**end**}
Predict(4)	= {**read**}
Predict(5)	= {**write**}
Predict(6)	= {**id**}
Predict(7)	= { **;** }
Predict(8)	= {**id**, **int**}
Predict(9)	= { **;** }
Predict(10)	= {**int**}
Predict(11)	= {**id**}

5) Nyní již můžeme zkonstruovat výslednou *LL*-tabulku. Tabulka má záhlaví sloupců popsané ***terminálními symboly*** a speciálním symbolem **$**, záhlaví řádků ***neterminálními symboly***. Jednotlivá políčka tabulky α[*A*, ***a***], kde *A* ∈ *N*, ***a*** ∈ *T* ∪ {$} potom vyplníme následujícím algoritmem:
   - α[*A*, ***a***] obsahuje pravidlo ***r***: ***A*** → *x*, pokud ***a*** ∈ *Predict*(***r***)
   - α[*A*, ***a***] je prázdné pro všechny ostatní případy

Poznámka k algoritmu:

- **Pokud by nějaké políčko α[*A*, *a*] mělo obsahovat více jak jedno pravidlo, pak daná gramatika NENÍ LL !!!**

Konstrukce LL-tabulky pro výše uvedený příklad:

- Konstrukce LL-tabulky pro sloupec označený terminálem begin:

	begin
<prog>	1
<st-list>	
<stat>	
<it-list>	
<item>	

Pravidla *r*: *A* → *x*, jejichž množiny *Predict*(*r*) obsahující begin:
1: **<prog> → begin <st-list> end** // *Predict*(1) = {**begin**}

- Konstrukce LL-tabulky pro sloupec označený terminálem id:

	id
<prog>	
<st-list>	2
<stat>	6
<it-list>	8
<item>	11

Pravidla *r*: *A* → *x*, jejichž množiny *Predict*(*r*) obsahující id:
2: **<st-list> → <stat> ; <st-list>** // *Predict*(2) = {**id**, **write**, **read**, **;** }
6: **<stat> → id := add <item> <it-list>** //*Predict*(6) = {**id**}
8: **<it-list> → <item> <it-list>** //*Predict*(8) = {**id**, **int**}
11: **<item> → id** //*Predict*(11) = {**id**}

- Konstrukce LL-tabulky pro sloupec označený terminálem int:

	int
<prog>	
<st-list>	
<stat>	
<it-list>	8
<item>	10

Pravidla *r*: *A* → *x*, jejichž množiny *Predict*(*r*) obsahující int:
8: **<it-list> → <item> <it-list>** //*Predict*(8) = {**id**, **int**}
10: **<item> → int** //*Predict*(10) = {**int**}

…

Ostatní sloupce tabulky by se sestavily analogicky.

Výsledná *LL*-tabulka pro výše uvedený příklad:

	begin	end	read	write	id	int	;	:=	add	$
<prog>	1									
<st-list>		3	2	2	2		2			
<stat>			4	5	6		7			
<it-list>					8	8	9			
<item>					11	10				

b) Nejprve si uvedeme algoritmus:

Algoritmus pro prediktivní syntaktickou analýzu používající LL-tabulku:

- Vlož na zásobník symboly **$*S***, kde ***S*** je startující neterminální symbol
- Hlavní cyklus
    - Nechť ***a*** je aktuální vstupní symbol, ***X*** je nejvrchnější symbol na zásobníku

        X = **$**: Pokud ***a*** = **$**, **úspěch syntaktické analýzy**;
        jinak **chyba!**

        X $\in$ *T*: Pokud ***X*** = ***a***, přečti symbol ***a*** ze vstupu a odstraň symbol ***a*** ze zásobníku;
        jinak **chyba!**

        X $\in$ *N*: Pokud *LL*-tabulka na souřadnicích [***X***, ***a***] obsahuje pravidlo ***r***: ***X*** $\rightarrow$ *x*,
        odstraň symbol ***X*** ze zásobníku a vlož na něj **reverzovaně** řetězec *x*;
        jinak **chyba!**
- Proveď další smyčku cyklu

Syntaktická analýza řetězce begin write int ; end :

Zásobník	**Vstup**	**Pravidlo**	**Odpovídající derivace**
$ <prog>	**begin write int ; end $**	1	**<prog> $\Rightarrow$ begin <st-list> end**
$ end <st-list> begin	**begin write int ; end $**		
$ end <st-list>	**write int ; end $**	2	**$\Rightarrow$ begin <stat> ; <st-list> end**
$ end <st-list> ; <stat>	**write int ; end $**	5	**$\Rightarrow$ begin write <item> ; <st-list> end**
$ end <st-list> ; <item> write	**write int ; end $**		
$ end <st-list> ; <item>	**int ; end $**	10	**$\Rightarrow$ begin write int ; <st-list> end**
$ end <st-list> ; int	**int ; end $**		
$ end <st-list> ;	**; end $**		
$ end <st-list>	**end $**	3	**$\Rightarrow$ begin write int ; end**
$ end	**end $**		
$	**$**		

ÚSPĚCH! Levý rozbor: 1 2 5 10 3