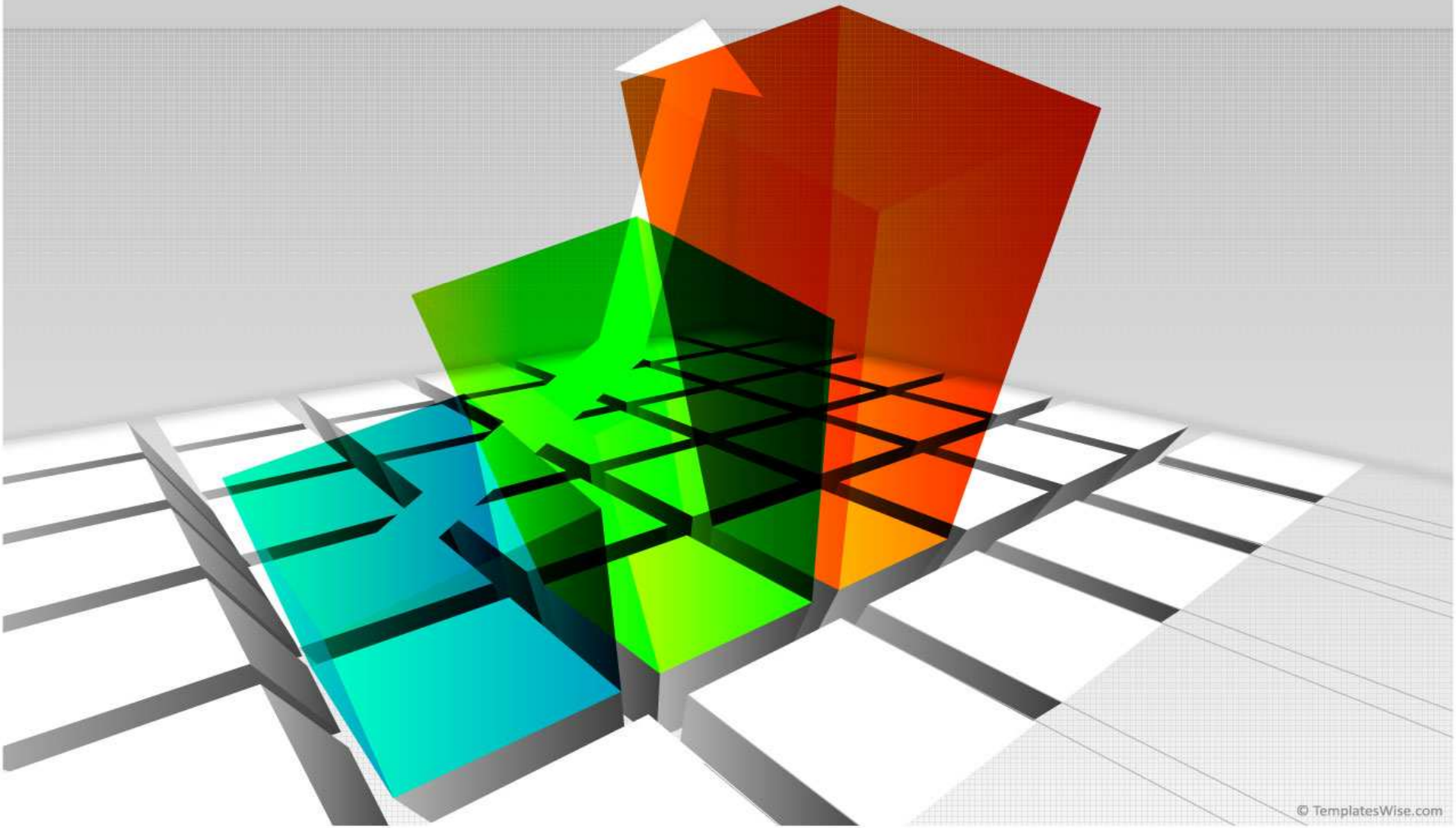
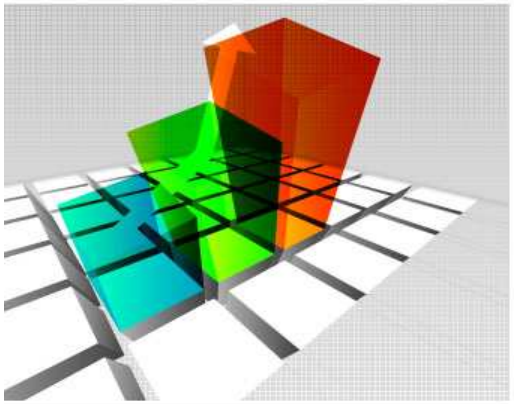
Конструкција на компајлер

Лабораториска вежба 6: Лексер



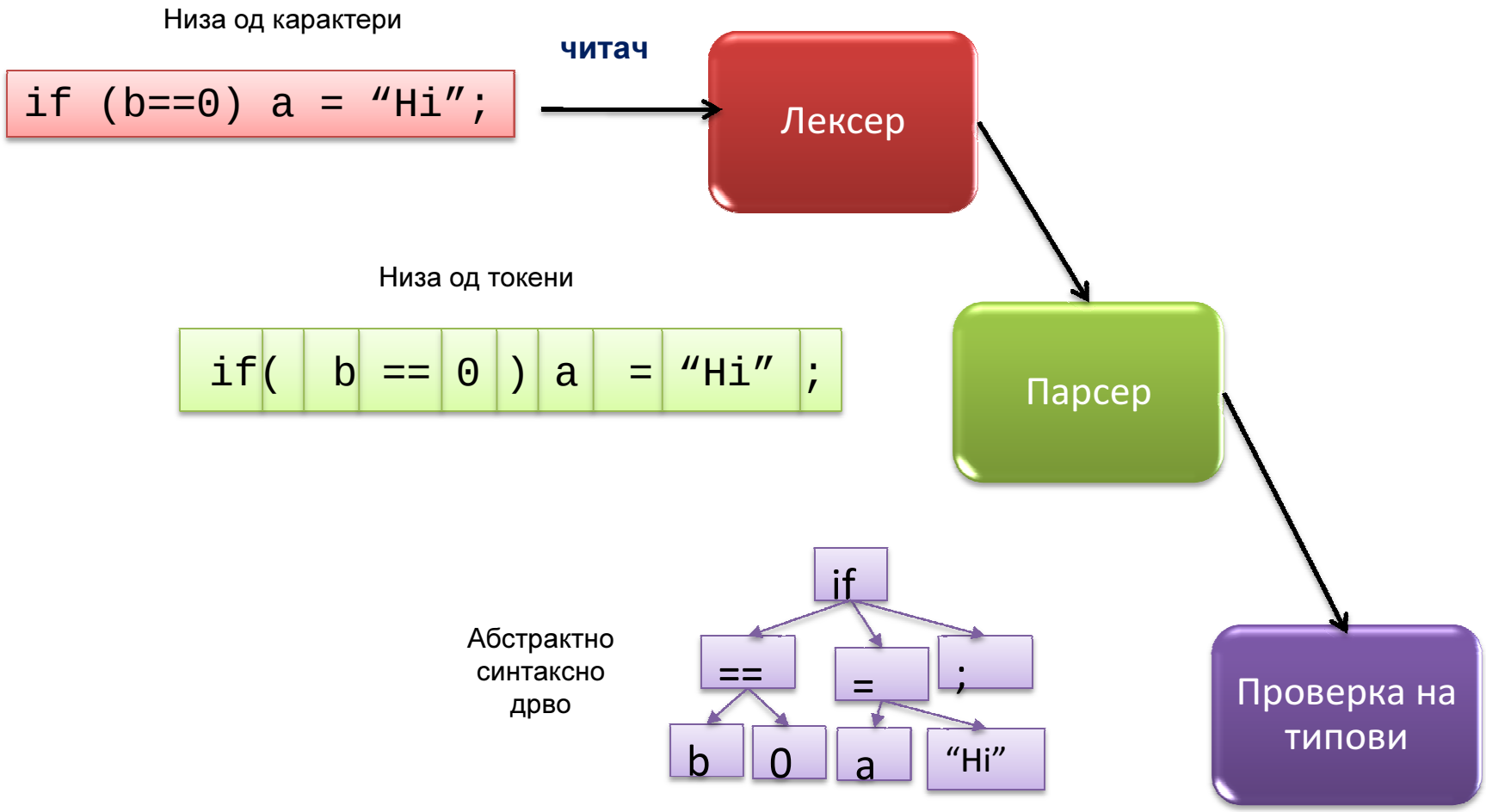


Лабораториска вежба 6

- **Фази на изградба на компајлер**
  - Лексичка анализа
  - Синтаксичка анализа
  - Семантичка анализа
  - Генерирање на код

Фази на изградба на компајлер

Што всушност се случува?



Читач (Scanner)

- Читач
  - Ни овозможува интефејс до низата карактери
  - На барање од лексерот генерира карактер по карактер
  - Го известува лескерот кога ке дојде до крајот на низата на карактери

Лексичка анализа

- Поими
  - Лексема
    - Најмал значаен ентитет во јазикот
      - Пр: клучни зборови, идентификатори, константи
  - Токен
    - Атомична единица од програмската синтакса
      - Пр: збор во споредба со реченица

Лексичка анализа

- Групирање на низа од карактери во лексеми
- Спарување на лексемите од јазикот и нивно класифицирање како токени
  - Имиња на променливи
  - Клучни зборови/резервирани зборови
  - Нумерички литерали
  - Стринг литерали

Лексичка анализа

- Токени и нивни типови

Лексема:	**Тип:**	**Токен:**
foo, x, listcount	ID	ID(foo), ID(x), ...
10.45, 3.14, -2.1	REAL	REAL(10.45), REAL(3.14), ...
;	SEMI	SEMI
(	LPAREN	LPAREN
50, 100	NUM	NUM(50), NUM(100)
if	IF	IF

Лексичка анализа

- Пример

x = (y + 4.0) ;

Лексичка анализа

ID(x) ASSIGN LPAREN ID(y) PLUS REAL(4.0) RPAREN SEMI

Конструкција на лексер

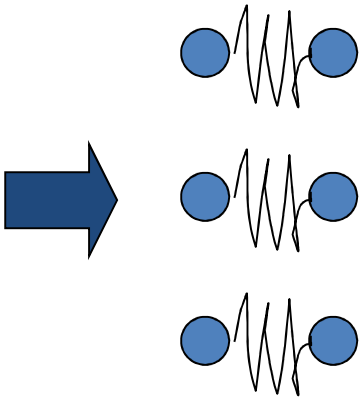
- Постапка
  1. Запишете ги регуларните изрази за влезниот јазик
  2. Изградете голем НДКА
  3. Креирајте ДКА за НДКА од чекор 2
  4. Минимизирајте го ДКА
  5. Искодирајте го ДКА

Конструкција на лексер

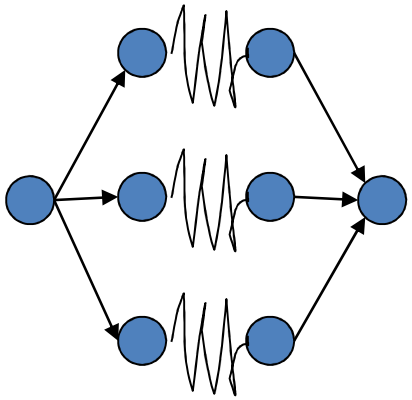
Спецификација

```
“if”
“while”
[a-zA-Z][a-zA-Z0-9]*
[0-9][0-9]*
(
)
```

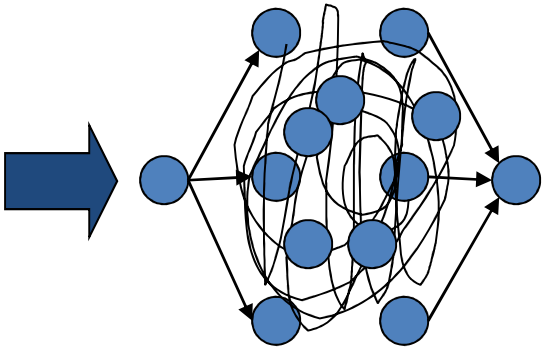
НДКА за секој RE



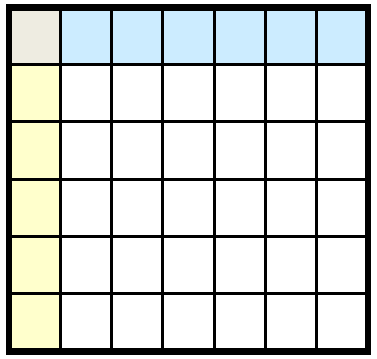
Голем НДКА



Голем ДКА



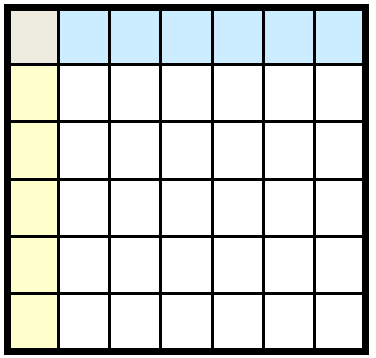
Табеларен приказ



Програмска имплементација на ДКА во функција на лексер

- Се чита знак по знак (од читачот)
- Со поаѓајќи од почетна состојба ја менуваме состојбата според табеларниот приказ на ДКА
  - Редици – состојби
  - Колони – знаци
  - Полињата во табелата –состојби
- Некои состојби предизвикуваат да се прати порака на парсерот.
  - Пораката е токенот кој е прочитан
- Кога нема букви за читање – или грешка или порака до парсерот дена нема веќе токени

Табеларен приказ



Резиме

- Лексер
  - Конвертира групи на карактери во токени
  - Генерира токен по токен на барање на парсерот
  - Го известува парсерот кога нема повеќе токени

Задолжение

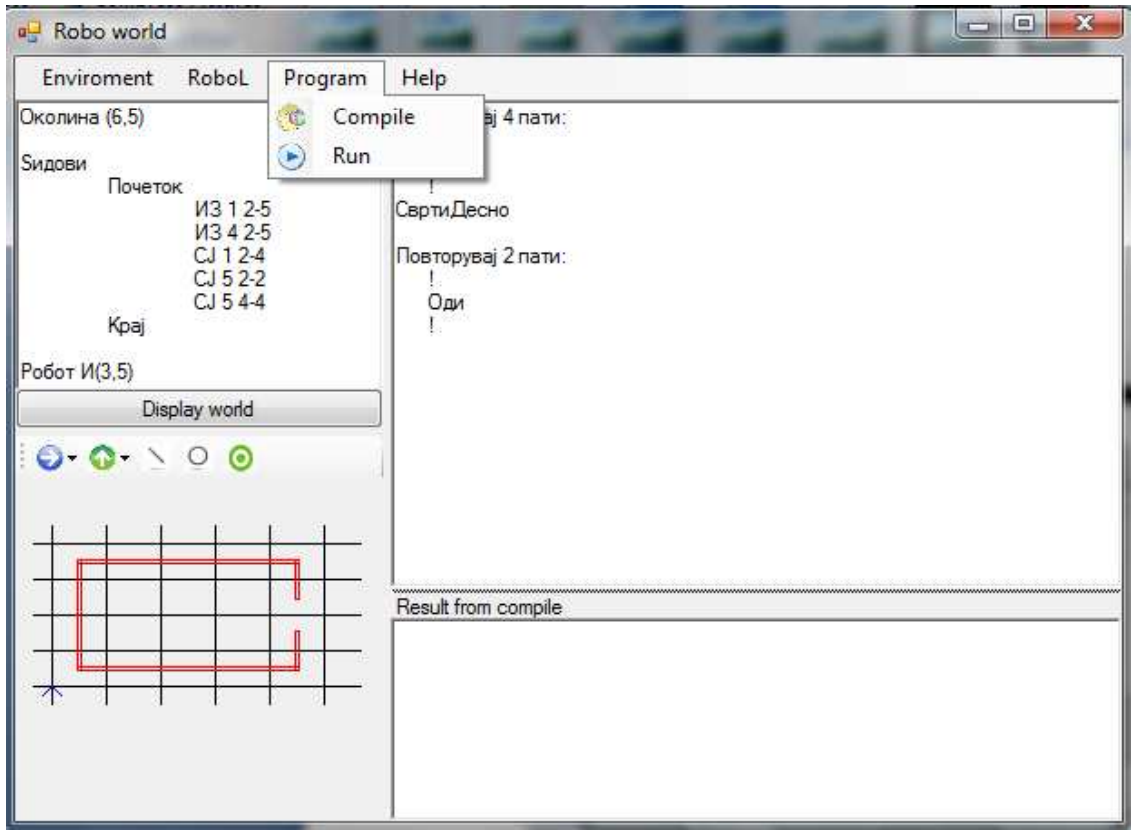
1. Да направите школка на апликацијата
2. Да дефинирате кои се токени во јазикот
3. Да го имплементирате лексерот
   - Се чита програма и се враќа низа од токени
   - Токени може да бидат ваши креирани класи
4. Компајлер за RoboEnv

1. Да направите школка на апликацијата

- За сите: околина за пишување, компајлирање и извршување на јазикот
- За оние кои треба да ја дефинираат околината со RoboEnv – пишување, компајлирање и извршување на јазикот (претставување на околината)
- За оние кои треба да ја дефинираат околината визуелно – креирање на околина, додавање и бришење на sидови и ознаки, поставување на роботот

1. Да направите школка на апликацијата

- Пример околина



2. Да дефинирате кои се токени во јазикот

- Во табела да се наведат сите токени (со или без доделена вредност)
  - Да се наведе кои се лексеми ги опфаќа токенот

3. Да го имплементирате лексерот

- За дадена внесена програма да ја најдете листата од токени која се добива со лексерот
- Да се одреди дали настанала лексичка грешка

4.Компајлер за RoboEnv

- За оние кои треба да ја дефинираат околината со RoboEnv
- RoboEnv е контектно-слободен јазик, но и регуларен
  - Зошто???
- Ако е регуларен може со помош на КА, да одредиме дали некоја програма припаѓа на јазикот
- Кои токени ги има јазикот???

4.Компајлер за RoboEnv

- За дадена програма во RoboEnv да одредите дали таа програма е правилно синтаксички напишана или не