

# Лінгвістичне забезпечення САПР

## Лекція 1

Стативка Юрій Іванович

[yurii.statyvka@gmail.com](mailto:yurii.statyvka@gmail.com)

# Лінгвістичне забезпечення САПР

Основна мета:

- теоретичні основи та
- практичні навичи  
розробки компіляторів мов високого рівня

# Організаційні аспекти

- 2 семестри (диф. залік + іспит)
- 1 лекція + 1 лабораторне заняття на тиждень
- 2 + 3 лабораторні роботи
- Курсова робота

# Мої очікування

## Ми працюємо разом!

### 1. Лекції

- Обговорення
- Питання
- Повідомлення/виступи

### 2. Лабораторні заняття

- ЛР: виконання/захист - самостійно/вчасно
- завдання/опитування, контрольні/самост. роботи

### 3. Курсова робота – самостійно, креативно

# Мої очікування

У кожного студента ВЖЕ Є

1. Улюблена мова програмування
2. Активний набір технологій/інструментів

# Оцінювання

У відповідності до шкали навчального плану

Особливості:

1. Якісне/самостійне/своєчасне виконання завдань –  
максимум = 95 балів
2. Виконання додаткового завдання –  
максимум = 97 балів
3. Додаткове завд. + конференція (доповідь/тези ) або  
стаття у науковому виданні –  
максимум = 100 балів

# Рекомендована література

1. Медведєва В.М. Транслятори: лексичний та синтаксичний аналізатори [Текст] : навч. посіб. / В.М. Медведєва, В.А. Третяк. – К. : НТУУ «КПІ», 2012. – 148с.
2. Медведєва В.М., Сліпченко В.Г. Основи побудови компіляторів Навчальний посібник.- К. : ІЗМН, 1999. - 104с.
3. Основи розробки трансляторів [Текст] : метод. вказівки до викон. курсових робіт для студ. напрямку підготов. 6.050103 «Програмна інженерія» та 6.050101 «Комп'ютерні науки» / уклад. : В.М. Медведєва, В.А. Третяк. – К. : НТУУ «КПІ», 2011. – 56 с.

# Рекомендована література

4. Ахо Альфред, Лам Моника, Сети Рави, Ульман Джеффри Д. Компиляторы: принципы, технологии и инструментарий, 2-е изд. : Пер. с англ. – М. : ООО „И.Д. Вильямс”, 2008. – 118
5. Джон Хопкрофт, Раджив Мотвани, Джеффри Ульман Введение в теорию автоматов, языков и вычислений, 2-е издание:Пер. с англ. - М.: Издательский до «Вильямс», 2002. – 528 с.
6. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Том 1, 2: - М.: Мир, 1978



# Рекомендована література

7. Фридл Дж. Регулярные выражения, 3е издание. – Пер. с англ. – СПб.: СимволПлюс, 2008. – 608 с.

8. Мозговой М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. – СПб.: Наука и техника, 2006. – 320 с.

9. ...

...

# Транслятори



# Транслятори

```
int sumcalc(int a, int b, int N)
{
    int i;
    int x, t, u, v;
    x = 0;
    u = ((a<<2)/b);
    v = 0;
    for(i = 0; i <= N; i++) {
        t = i+1;
        x = x + v + t*t;
        v = v + u;
    }
    return x;
}
```

Транслятор  
(компілятор,  
інтерпретатор\*)

```
sumcalc:
    xorl    %r8d, %r8d
    xorl    %ecx, %ecx
    movl    %edx, %r9d
    cmpl    %edx, %r8d
    jg      .L7
    sall    $2, %edi
.L5:
    movl    %edi, %eax
    cltd
    idivl    %esi
    leal    1(%rcx), %edx
    movl    %eax, %r10d
    imull    %ecx, %r10d
    movl    %edx, %ecx
    imull    %edx, %ecx
    leal    (%r10,%rcx), %eax
    movl    %edx, %ecx
    addl    %eax, %r8d
    cmpl    %r9d, %edx
    jle     .L5
.L7:
    movl    %r8d, %eax
    ret
```

Правила мови  
(алфавіт,  
синтаксис,  
семантика, ...)

Програма  
вхідною мовою  
(source language)

Розпізнавання та  
генерування  
ЕЛЕМЕНТІВ  
програми

Транслятор  
(компілятор,  
інтерпретатор\*)

Правила мови  
(алфавіт,  
синтаксис,  
семантика, ...)

Програма  
цільовою мовою  
(target language)

- 1.Формальні мови
- 2.Формальні граматики
- 3.Автомати
- 4.Графи, дерева, ...

# Передбачається використання

1. Для розробки власного транслятора – ваші улюблені мови програмування та технології –
2. Для вивчення формальних мов, граматик та автоматів – JFLAP. <http://www.jflap.org/>
3. Для побудови синтаксичних діаграм в розширеній нотації Бекуса-Наура – EBNF Visualizer. <https://sourceforge.net/projects/ebnfvisualizer/>
4. Для автоматизації розробки транслятора – генератори компіляторів/парсерів -YACC, LEX, ANTLR, Roslyn, Coco/R, ....

# Функція та структура транслятора



*Програма (character stream)*

Лексичний аналіз

Lexical Analyzer (Scanner)



*Потік токенів/лексем (token stream)*

# Lexical Analyzer (Scanner)

2	3	4		*		(	1	1		+	-	2	2	)							
---	---	---	--	---	--	---	---	---	--	---	---	---	---	---	--	--	--	--	--	--	--

Num(234) mul\_op lpar\_op Num(11) add\_op Num(-22) rpar\_op

18..23 + val#ue

Not a number

Variable names cannot have '#' character

# Функція та структура транслятора



*Програма (character stream)*

Лексичний аналіз

Lexical Analyzer (Scanner)



*Потік токенів/лексем (token stream)*

Синтаксичний аналіз

Syntax Analyzer (Parser)

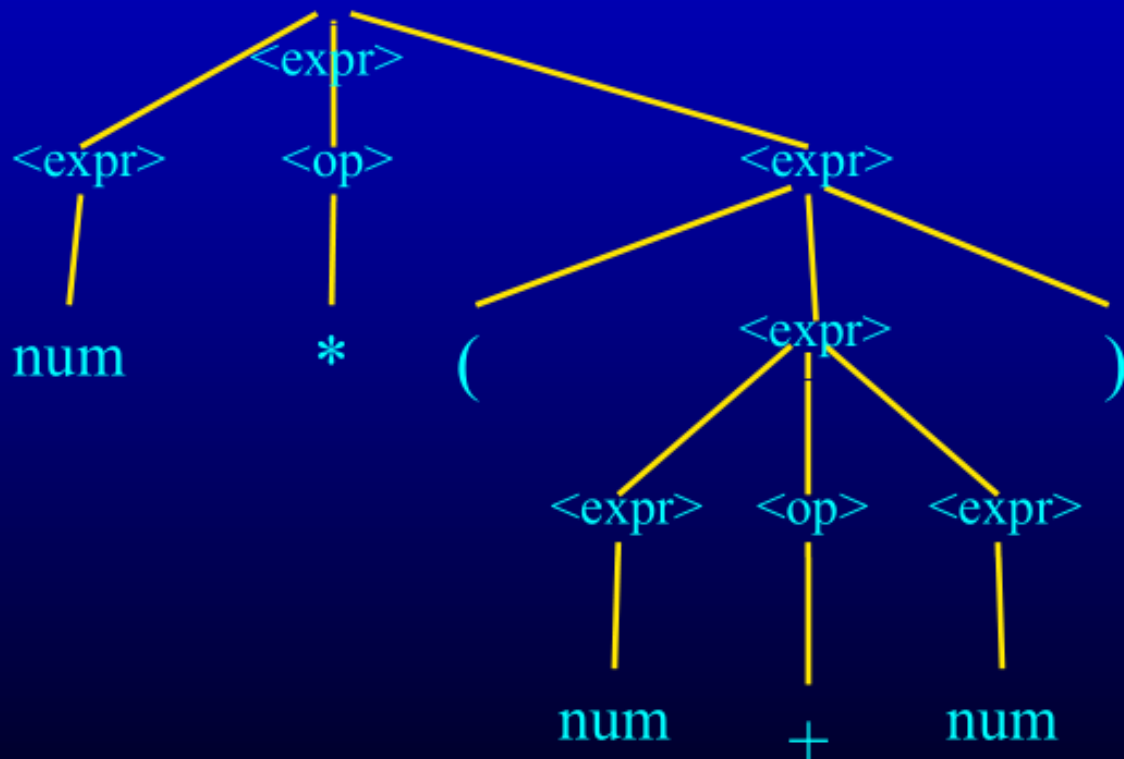


*Синтаксичне дерево (Parse Tree)*



# Syntax Analyzer (Parser)

num '\*' '(' num '+' num ')'



# Syntax Analyzer (Parser)

```
int * foo(i, j, k))
```

```
    int i;
```

```
    int j;
```

```
{
```

```
    for(i=0; i j) {
```

```
    fi(i>j)
```

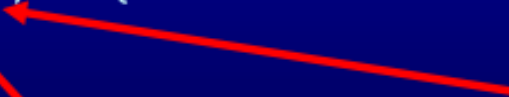
```
        return j;
```

```
}
```

Extra parentheses



Missing increment



Not an expression



Not a keyword



# Функція та структура транслятора



*Програма (character stream)*

Лексичний аналіз

Lexical Analyzer (Scanner)



*Потік токенів/лексем (token stream)*

Синтаксичний аналіз

Syntax Analyzer (Parser)



*Синтаксичне дерево (Parse Tree)*

Семантичний аналіз

Semantic Analyzer



*Intermediate Representation (IR)*

# Semantic Analyzer

```
int * foo(i, j, k)
```

```
    int i;
```

```
    int j;
```

```
{
```

```
    int x;
```

```
    x = x + j + N;
```

```
    return j;
```

```
}
```

Type not declared

Mismatched return type

Uninitialized variable used

Undeclared variable

# Функція та структура транслятора



*Програма (character stream)*

Лексичний аналіз

Lexical Analyzer (Scanner)



*Потік токенів/лексем (token stream)*

Синтаксичний аналіз

Syntax Analyzer (Parser)



*Синтаксичне дерево (Parse Tree)*

Семантичний аналіз

Semantic Analyzer



*Intermediate Representation (IR)*

Оптимізація коду

Code Optimizer



*Optimized IR*

# Optimizer

```
int sumcalc(int a, int b, int N)
{
    int i;
    int x, y;
    x = 0;
    y = 0;
    for(i = 0; i <= N; i++) {
        x = x+4*a/b*i+(i+1)*(i+1);
        x = x + b*y;
    }
    return x;
}
```



```
int sumcalc(int a, int b, int N)
{
    int i;
    int x, t, u, v;
    x = 0;
    u = ((a<<2)/b);
    v = 0;
    for(i = 0; i <= N; i++) {
        t = i+1;
        x = x + v + t*t;
        v = v + u;
    }
    return x;
}
```

# Функція та структура транслятора



*Програма (character stream)*

Лексичний аналіз

Lexical Analyzer (Scanner)



*Потік токенів/лексем (token stream)*

Синтаксичний аналіз

Syntax Analyzer (Parser)



*Синтаксичне дерево (Parse Tree)*

Семантичний аналіз

Semantic Analyzer



*Intermediate Representation (IR)*

Оптимізація коду

Code Optimizer



*Optimized IR*

Генерування коду

Code Generator



*Assembly code*

# Code Generator

```
int sumcalc(int a, int b, int N)
{
    int i;
    int x, t, u, v;
    x = 0;
    u = ((a<<2)/b);
    v = 0;
    for(i = 0; i <= N; i++) {
        t = i+1;
        x = x + v + t*t;
        v = v + u;
    }
    return x;
}
```



```
sumcalc:
    xorl    %r8d, %r8d
    xorl    %ecx, %ecx
    movl    %edx, %r9d
    cmpl    %edx, %r8d
    jg      .L7
    sall    $2, %edi
.L5:      movl    %edi, %eax
    cltd
    idivl    %esi
    leal    1(%rcx), %edx
    movl    %eax, %r10d
    imull    %ecx, %r10d
    movl    %edx, %ecx
    imull    %edx, %ecx
    leal    (%r10,%rcx), %eax
    movl    %edx, %ecx
    addl    %eax, %r8d
    cmpl    %r9d, %edx
    jle     .L5
.L7:      movl    %r8d, %eax
    ret
```



# Функція та структура транслятора



*Програма (character stream)*

Лексичний аналіз

Lexical Analyzer (Scanner)



*Потік токенів/лексем (token stream)*

Синтаксичний аналіз

Syntax Analyzer (Parser)



*Синтаксичне дерево (Parse Tree)*

Семантичний аналіз

Semantic Analyzer



*Intermediate Representation (IR)*

Оптимізація коду

Code Optimizer



*Optimized IR*

Генерування коду

Code Generator



*Assembly code*

# Завдання

1. Опитування за розділом 1 з [1]. Завдання для самоконтролю.
2. Завантажити JFLAP з <http://www.jflap.org/>
3. Створити прості автомати засобами JFLAP. Виконати маніпуляції у відповідності до розділу *JFLAP Tutorial > Finite Automata > Construct and Run*, див. <http://www.jflap.org/tutorial/>
4. Ознайомитись з розділом 1 з книги Susan H. Rodger and Thomas W. Finley “JFLAP: An Interactive Formal Languages and Automata Package”. Книга доступна за адресою <http://www.jflap.org/jflapbook/jflapbook2006.pdf>

Приклад оптимізації від MIT OpenCourseWare

<http://ocw.mit.edu>

6.035 Computer Language Engineering

Spring 2010

Слайди 48 - 76