Вимоги до виконання лабораторних та розрахунково-графічної робіт з дисципліни

“Інженерія програмного забезпечення 1. Основи проектування трансляторів”

2 семестр 2017/2018

Формальні вимоги до вмісту протоколу.

- ☐ Титульний лист (шаблони титульних листів додаються).

- ☐ Індивідуальне завдання згідно варіанту:
  - ☐ Номер варіанту (Так, ще раз. Так, і на титульному листі також обов'язково).
  - ☐ Граматика за варіантом в гарно читабельному вигляді.

- ☐ Граф автомату (для лабораторної роботи №1)
  - ☐ ***Граф в точності повинен відповідати граматиці за варіантом***, тобто повинен містити стани для розбору всіх можливих за вашим варіантом токенів і не містити станів, які не потрібні в даному варіанті.
  - ☐ Граф повинен містити гілку станів розбору коментаря. Необхідно добре розуміти як ця гілка станів працює, оскільки одним із завдань може бути намалювати гілку станів розбору коментаря і розказати як вона працює.
  - ☐ Граф **не** є скріншотом (чи будь-якою іншою копіпастою) з методички Рекомендація: наполегливо рекомендується розібратися із http://graphviz.org/ (використовуйте dot). Надзвичайно корисна річ, в майбутньому не раз знадобиться. Якщо лінь розбиратися з дотом, можна використати https://www.draw.io/

- ☐ Таблиця переходів машини Кнута відповідно до граматики за варіантом (для РГР, у випадку, якщо заданий метод синтаксичного аналізу — машина Кнута).

- ☐ Код програми
  - ☐ Моноширинний шрифт, не більше 10 розміру, можна 8 (залежить від шрифту). Якщо код не широкий, використайте дві колонки (https://goo.gl/wy9ub6)
  - ☐ Повинен містити **весь** код програми

- ☐ Тести
  - ☐ Тест складається із коду мовою SIGNAL та результату роботи програми лабораторної або розрахунково-графічної роботи.
  - ☐ Тести повинні бути двох категорій:
    - ☐ тести, що демонструють правильну роботу програми при правильній вхідній програмі (True-тести).
    - ☐ тести, що демонструють правильну роботу програми при неправильній вхідній програмі (False-тести), тобто виведення правильного повідомлення про помилку з вказанням правильного місця (рядок, колонка) цієї помилки.
  - ☐ True-тести повинні покривати різні варіанти правильного використання конструкцій граматики свого варіанту.
  - ☐ False-тести повинні покривати всі можливі типи помилок у конструкціях лексики, граматики та семантики свого варіанту.
  - ☐ ***На захист лабораторних та розрахунково-графічної роботи приходити з вже підготовленими тестами обох категорій, що повністю покривають лексику (ЛР №1), граматику (РГР) та семантику (ЛР №2) свого варіанту. Ці тести повинні бути підготовлені у різних файлах вже у готовому вигляді, а не коригуватись один і той же файл під час захисту. Кількість тестів залежить від варіанту граматики та її семантики.***
  - ☐ У звітах достатньо надрукувати по 3-5 тестів кожної категорії (True-тести, False-тести) для кожної лабораторної та розрахунково-графічної роботи.
  - ☐ Див. також "Вказівки щодо написання тестів" нижче у цьому файлі.
  - ☐ У звітах не повинно бути скріншотів вікна консолі з чорним фоном. Треба або копіювати текст (бажано), або інвертувати колір скріншоту (https://goo.gl/ki6Amb). В разі використання скріншоту вирізайте лише мінімальну область з результатом. Для цього можна використати вбудовану у Windows7+ утиліту Snipping tool.

Приклади оформлення тестів для лабораторної №1.

1. True-тест

```
PROGRAM SIG01;
BEGIN
END.
```

```
1   1   1001  PROGRAM
1   9   2001  SIG01
1   14  501   ;
2   1   1002  BEGIN
3   1   1003  END
3   4   502   .
```

Плюс роздруківка сформованих інформаційних таблиць

2. False-тест

```
PROGRAM SIG01;
BEGIN ?
END.
```

```
1   1   1001  PROGRAM
1   9   2001  SIG01
1   14  501   ;
2   1   1002  BEGIN
```

Lexer: Error (line 2, column 7): Illegal symbol ‘?’

Плюс роздруківка інформаційних таблиць, сформованих до моменту знаходження помилки.

Приклади оформлення тестів для РГР.

1. True-тест

```
PROGRAM SIG01;
BEGIN
END.
```

```
<signal-program>
..<program>
....1001  PROGRAM
....<procedure-identifier>
......<identifier>
........2001  SIG01
....501  ;
....<block>
......<declarations>
........<variable-declarations>
..........<empty>
......1002  BEGIN
......<statements-list>
........<empty>
......1003  END
....502  .
```

2. False-тест

```
PROGRAM SIG01;
BEGUN
END.
```

```
<signal-program>
..<program>
....1001  PROGRAM
....<procedure-identifier>
......<identifier>
........2001  SIG01
....501  ;
```

Parser: Error (line 2, column 1): Keyword ‘BEGIN’ expected

Загальні вимоги до програми будь-якої лабораторної роботи чи РГР

1. Заборонено використовувати мову програмування JavaScript, а також всі її діалекти та похідні.
   Завдячуйте за цю заборону старшим курсам, які чомусь вважали, що ми не знаємо JS, і 95% лаб на JS були відвертим “палевом”. Використання JS дозволяється в індивідуальному порядку, і лише за умови обирання ускладненого варіанту.

2. Заборонено використовувати вбудовані у мови програмування та бібліотечні засоби роботи з регулярними виразами.

3. **Backend лабораторних робіт та РГР** (тобто алгоритмічна частина лексичного аналізатора, синтаксичного аналізатора та генератора коду) **повинен бути строго відокремлений від frontend-у** (інтерфейсної частини роботи з програмою) **і знаходитись у окремих модулях/класах тощо**.
   Функції виведення рядка лексем, дерева розбору та інформаційних таблиць на друк є частиною frontend-у, а не backend-у. Вони повинні викликатись один раз після того як структури даних вже остаточно сформовані, а не бути переплетеними за принципом “лапші” з основною логікою в алгоритмічній частині.

4. Кожна лабораторна робота повинна бути виконана у вигляді незалежної програми. Кожна наступна лабораторна може містити в собі код попередньої, але не навпаки.

5. Повідомлення про помилки повинні бути якнайбільш змістовними, містити в собі всі дані про помилку, які є на момент її виникнення, а також тип помилки.
   Наприклад:
   ```
   “Lexer: Error (line 2, column 1): Illegal character ‘^’ detected”
   “Parser: Error (line 10, column 15): ‘<identifier>’ expected but ‘;’ found”
   “Code Generator: Error (line 2, column 1): Incompatible types of variables ‘i:integer’ ‘s:string’ in assignment”
   ```

6. На етапі лексичного аналізу (лабораторна робота №1) для кожного токена у рядку токенів (лексем) зберігається позиція їх знаходження, тобто номер рядка та колонки першого символу токена, так, як показано на лекційному рисунку лексичного аналізатора.
   Позиції нумеруються від одиниці, а не від нуля.
   Синтаксичний аналізатор та генератор коду також повинні використовувати ці позиції.
   Зверніть увагу на те, що колонка першого символу токена — це не те ж саме, що порядковий номер токена в межах одного рядка.

Вказівки щодо написання тестів та інші вимоги, специфічні для конкретної роботи

Дані вказівки не обов’язково означають “написати по тесту на вказівку”. Це неповний список ситуацій, в коректності яких треба впевнитися перед тим, як намагатися здати роботу. Можливо це наштовхне вас на думки як ще можна протестувати лабораторну. Вам вирішувати, чи необхідно писати окремий тест на ту чи іншу ситуацію.

Лексичний аналізатор

- **Читання із вихідного файлу відбувається посимвольно**
- Позиції коректно визначаються для всіх токенів (перевіряти в текстовому редакторі, наприклад, Notepad). Позиція кожного токена, збережена в рядку токенів, повинна співпадати з фактичною позицією цього токена в текстовому редакторі.
- Позиції токенів після багаторядкового коментаря також повинні бути коректними. Наприклад, в наступних двох рядках колонка токена “token” є різною.
  - The last line of `comment *)token`
  - The last line of `comment *) token`
- Різні варіанти коментарів повинні оброблюватись коректно, наприклад
  - Порожні коментарі (**) (* *)
  - (* багаторядковий
    коментар *)
  - (*****) (*(()*())*) (*;.:*)
  - Незакриті коментарі
  - Коментарі із забороненими символами: всередині коментаря заборонені символи *допускаються*
- Правильна обробка заборонених символів: повинна друкуватись помилка щодо знаходження забороненого символа
- Коректна обробка комбінацій роздільників, таких як
  ...;;:(()()());;:=:=:::=;;:.
  Наприклад, комбінація роздільників
  :::==
  повинна бути розібрана як дві двокрапки, одне присвоєння і один знак рівності.

Синтаксичний аналізатор

- Зміст тестів відповідає завданню за варіантом
- Повинен бути тест, що містить в собі всі можливі конструкції граматики (максимальний тест)
- Дерево розбору повинно містити в собі
  - Всі без виключення нетермінали, по яким розібрався вхідний код
  - Тобто, якщо у вас відсутній який-небуть список (тобто, за граматикою може бути <empty>), то необхідно показати в дереві весь шлях до <empty> включно
- Правила граматики, які містять в собі альтернативу <empty> , по факту дозволяють використовувати порожні конструкції