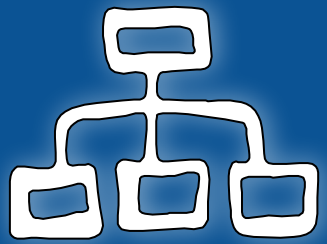
Алгоритмы и Алгоритмические Языки (C++)

Семинар #7:

1. Архитектура парсера языка эмулятора.
2. Алгоритм распознавания регулярного выражения.
3. Представление команд эмулятора.
4. Реализация динамического полиморфизма в C.

27.02.2024

Архитектура парсера регулярного языка



Задание языка для обработки

Рассматриваемый язык – сложное регулярное выражение.

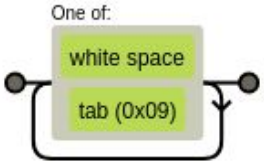
Базовые элементы языка (в форме Бэкуса-Наура):

```
<space-seq>   ::= “[ \t]+”
<newline-seq> ::= “\n+”
<cmd-id>      ::= “BEGIN”|”END”|”PUSH”|”POP”|”PUSHR”|”POPR”
<cmd-name>    ::= [<space-seq>] <command-id>
<reg-id>      ::= “AX”|“BX”|”CX”|”DX”|”EX”|”FX”|”PC”
<reg-name>    ::= <space-seq> <reg-id>
<value>       ::= "(\\+|-)?(0|[1-9][0-9]*)"
```

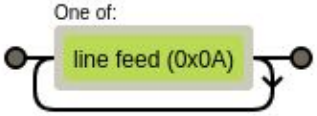
Пример парсера – 20_parser.

Парсеры базовых элементов языка

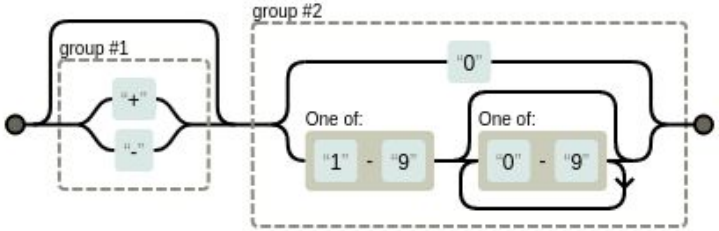
<space-seq> ::= “[\t]+”



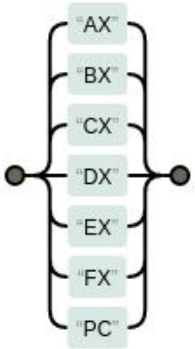
<newline-seq> ::= “\n+”



<value> ::= "(\\+|-)?(0|[1-9][0-9]*)"



<reg-id> ::= “AX”|“BX”|”CX”|”DX”|”EX”|”FX”|”PC”



Задание языка для обработки

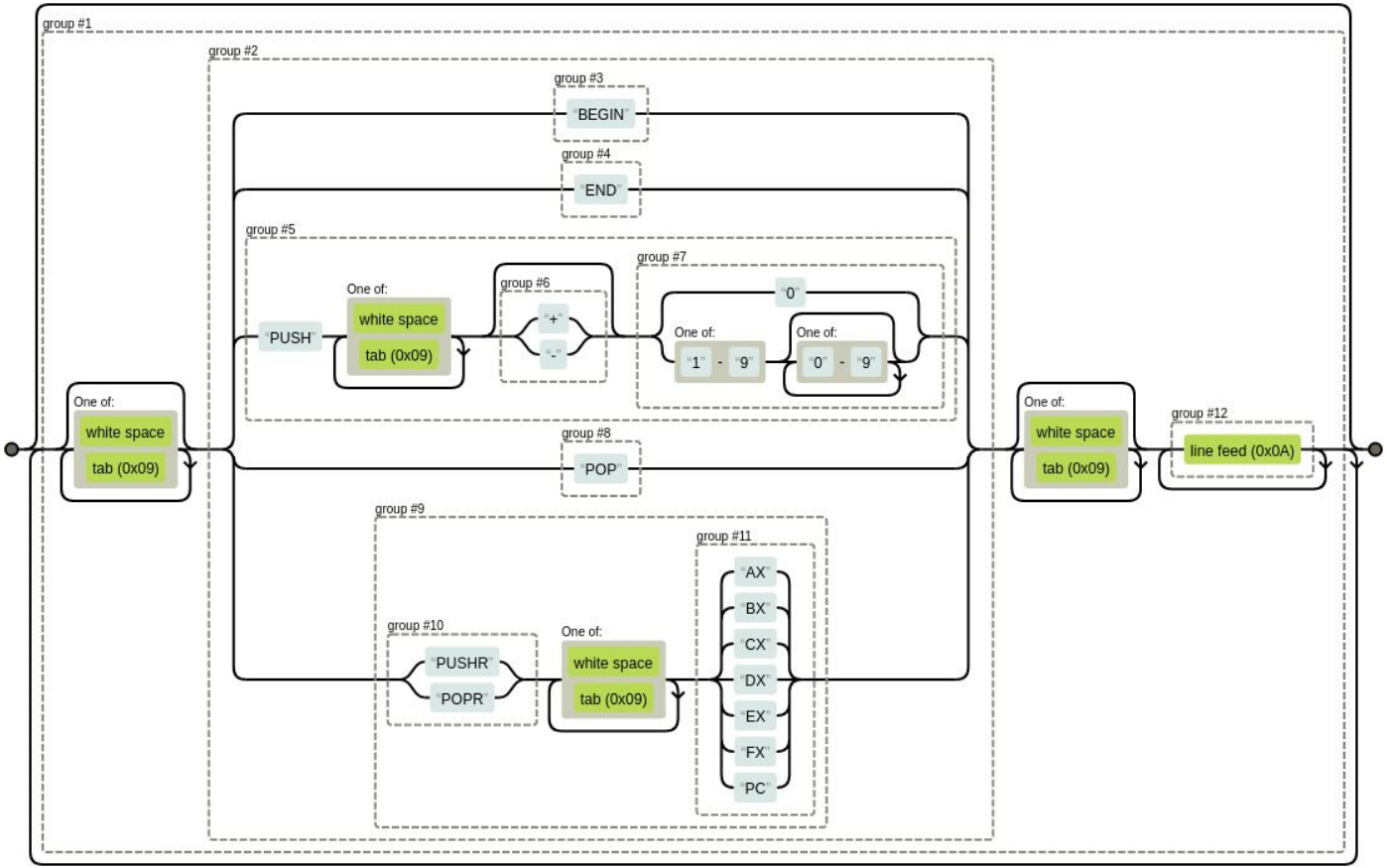
Команды языка:

```
<begin> ::= [<space-seq>] “BEGIN”
<end>   ::= [<space-seq>] “END”
<push>  ::= [<space-seq>] “PUSH” <space-seq> <value>
<pop>   ::= [<space-seq>] “POP”
<pushr> ::= [<space-seq>] “PUSHR” <space-seq> <reg-name>
<popr>  ::= [<space-seq>] “POPR” <space-seq> <reg-name>
```

Полное выражение для языка:

```
<command>      ::= <begin>|<end>|<push>|<pop>|<pushr>|<popr>
<command-line> ::= <command> [<space-seq>] <newline-seq>
<program>      ::= <command-line>*
```

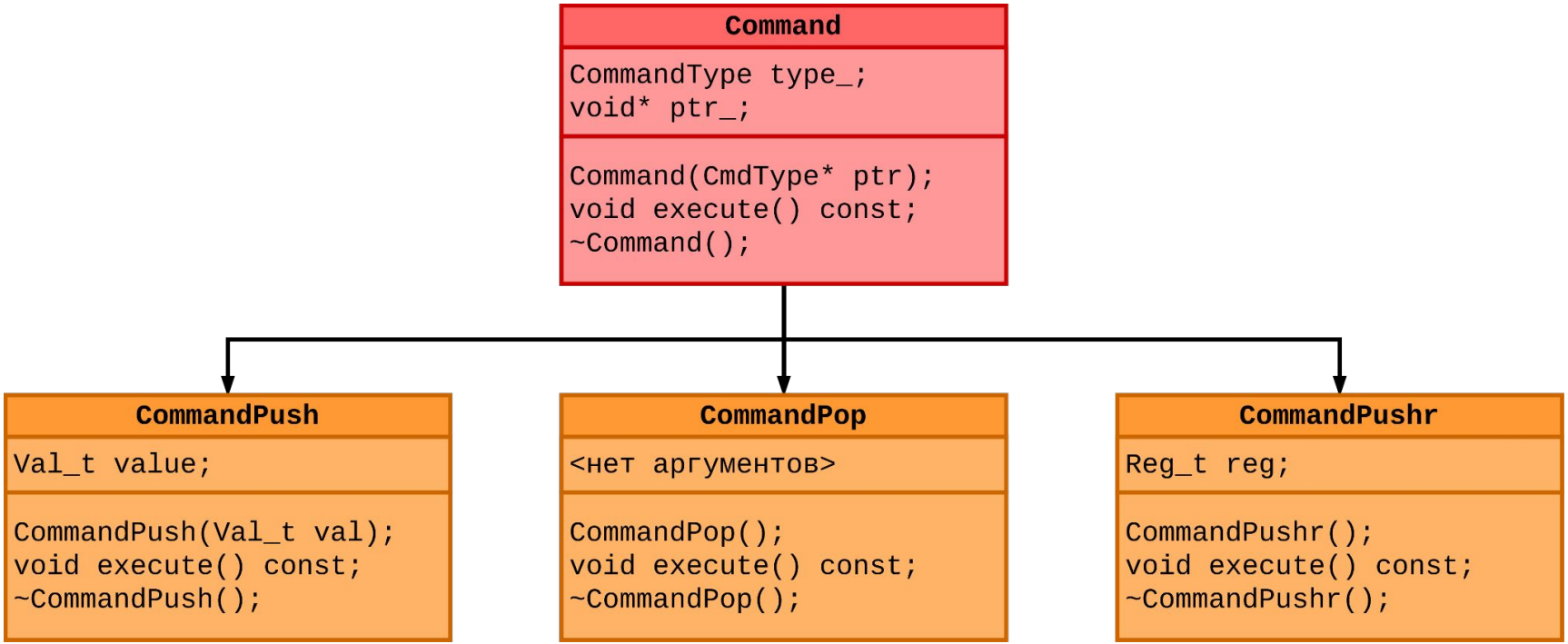
Архитектура парсера языка эмулятора



HSE University

Команды эмулятора. Реализация динамического полиморфизма в Си

Команды эмулятора



Ручная диспетчеризация при вызове `~Command()`, `execute()`.

Вопросы?

Красивые иконки взяты с сайта handdrawngoods.com