

## Завдання 1

Описати мову, яка описується регулярним виразом

3	$((\epsilon a) \cdot (c,b))^+$
---	--------------------------------

```
#include <iostream>
```

```
#include <regex>
```

```
bool is_valid_string(const std::string& str) {  
    std::regex re("((ε|a)·(c,b))+");  
    return std::regex_match(str, re);  
}
```

```
int main() {  
    std::string input;  
    std::cout << "Введіть рядок: ";  
    std::cin >> input;  
    if (is_valid_string(input)) {  
        std::cout << "Рядок належить мові.\n";  
    } else {  
        std::cout << "Рядок не належить мові.\n";  
    }  
  
    return 0;  
}
```

```
/tmp/QP9vuk0xVa.o
```

```
Введіть рядок: sad
```

```
Рядок не належить мові.
```

```
=== Code Execution Successful ===
```

## Завдання 2

Написати регулярні визначення для наступних мов:

3	Усі стрічки з цифр, які містять цифри за зростанням
---	---

```
#include <iostream>

#include <regex>

bool is_valid_string(const std::string& str) {
    std::regex re("^[0-9]+$");
    return std::regex_match(str, re);
}

int main() {
    std::string input;
    std::cout << "Введіть рядок: ";
    std::cin >> input;
    if (is_valid_string(input)) {
        std::cout << "Рядок належить мові.\n";
    } else {
        std::cout << "Рядок не належить мові.\n";
    }
    return 0;
}
```

```
/tmp/DFtvxIWbdf.o
```

```
Введіть рядок: 12345
```

```
Рядок належить мові.
```

```
=== Code Execution Successful ===
```

### Завдання 3

Використовуючи стандарт мови написати вхідну абетку для мови програмування, лексем ключових слів (до 20), лексеми односимвольних і багатосимвольні роздільників, лексеми арифметичних операцій, лексеми логічних і бітових операцій):

#### 3. Python

```
import re
```

```
KEYWORDS = [
```

```
    "and", "as", "assert", "break", "class", "continue", "def", "del",
    "elif", "else", "except", "finally", "for", "from", "global", "if",
    "import", "in", "is", "lambda", "not", "or", "pass", "print", "raise",
    "return", "try", "while", "with", "yield",
```

```
]
```

```
SINGLE_DELIMITERS = ["(", ")", "[", "]", "{", "}", ";", ":", "?", "!", "=",
    "+", "-", "*", "/", "%", "&", "|", "^", "~", "<", ">", "<=", ">=", "==", "!=", "<>"]
```

```
MULTI_DELIMITERS = [("*/", re.compile(r"^\*.*"))]
```

```
ARITHMETIC_OPERATORS = ["+", "-", "*", "/", "//", "%", "**"]
```

```
LOGICAL_OPERATORS = ["and", "or", "not"]
```

```
BITWISE_OPERATORS = ["&", "|", "^", "~", "<<", ">>"]
```

```
def tokenize(source_code):
```

```
    tokens = []
```

```
    for match in re.finditer(r"^[^\s]+", source_code):
```

```

token = match.group()
if token in KEYWORDS:
    tokens.append((token, "keyword"))
elif token in SINGLE_DELIMITERS:
    tokens.append((token, "delimiter"))
elif token in MULTI_DELIMITERS:
    for delimiter, regex in MULTI_DELIMITERS:
        if regex.match(token):
            tokens.append((delimiter, "delimiter"))
            break
elif token in ARITHMETIC_OPERATORS:
    tokens.append((token, "arithmetic_operator"))
elif token in LOGICAL_OPERATORS:
    tokens.append((token, "logical_operator"))
elif token in BITWISE_OPERATORS:
    tokens.append((token, "bitwise_operator"))
else:
    tokens.append((token, "identifier"))
return tokens


```

#Приклад використання словника



```

programmer_input = "x = 2 + 3 * y"
tokens = tokenize(programmer_input)
print(tokens)

```



```
[( 'x', 'identifier'), ('=', 'delimiter'), ('2', 'identifier'), ('+', 'delimiter'), ('3', 'identifier'), ('*', 'delimiter'), ('y', 'identifier')]
```



```
>_ ** Process exited - Return Code: 0 **  
Press Enter to exit terminal
```

