**5-зертхана: Классикалық есептер үшін рекурсивті функцияларды орындау (2 апта)**

**Мақсат:**

Классикалық алгоритмдік және математикалық есептерді шешу үшін Python тілінде рекурсивті функцияларды үйреніңіз және қолданыңыз. Жұмыстың мақсаты студенттердің рекурсивті кодты жазу, оның принциптері мен шектеулерін түсіну, сонымен қатар рекурсивті алгоритмдерді талдау және оңтайландыру дағдыларын дамыту болып табылады.

Тапсырмалар:

1. Рекурсия негіздері:

- Рекурсия ұғымын және оның итеративті тәсілден айырмашылығын түсіну.

- Рекурсивті функциялардың негізгі мысалдарын оқу.

2. Рекурсивті функциялардың дамуы:

- Классикалық рекурсивті есептерді орындау, мысалы, факториалды, Фибоначчи сандарын, екілік іздеуді және т.б.

- Итерациялық әдістермен салыстырғанда рекурсивті функциялардың тиімділігін талдау.

3. Қоңыраулар стегі мен рекурсиялық шектеулерді түсіну:

- Python тіліндегі шақыру стек механизмін және оның рекурсивті функцияларға әсерін зерттеу.

- Қоңыраулар стекінің толып кету мәселесін анықтау және болдырмау.

4. Рекурсивті функцияларды оңтайландыру:

- Рекурсивті қоңырауларды оңтайландыру үшін есте сақтау және динамикалық бағдарламалау әдістерін пайдалану.

- оңтайландырылған рекурсивті функциялардың өнімділігін олардың оңтайландырылмаған нұсқаларымен салыстыру.

5. Практикалық есептердегі рекурсияның қолданылуы:

- Ағаштарды, графиктерді және басқа деректер құрылымдарын аралау сияқты рекурсивті тәсілді қажет ететін күрделірек мәселелердің шешімдерін әзірлеу.

Зертханалық жұмыстың маңыздылығы:

Бұл зертхана студенттерге рекурсияны бағдарламалаудағы қуатты құрал ретінде түсінуді тереңдетуге көмектеседі. Итерациялық әдістерді қолдану арқылы шешуі қиын немесе тиімсіз есептерді шешу үшін жиі рекурсивті әдістер қолданылады. Рекурсияны меңгеру программисттің құралдар жиынтығын кеңейтіп, алгоритмдік есептердің кең ауқымын тиімді шешуге мүмкіндік береді.

**Жеке тапсырмалар:**

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз). Студенттерге рекурсияға, деректер құрылымдарын айналып өтуге және рекурсивті функцияларды оңтайландыруға көмектесетін есептер

1. Факториалды есептеу

- Санның факториалын есептеу үшін рекурсивті функцияны орындау.

2. Фибоначчи сандары

- \(n\)-ші Фибоначчи санын есептеу үшін рекурсивті функцияны жазыңыз.

3. Тізім элементтерінің қосындысы

- Тізімдегі элементтердің қосындысын табу үшін рекурсивті функцияны орындаңыз.

4. Ағаштарды аралау

- Екілік ағашты рекурсивті айналдыру және оның элементтерін шығару функциясын жазыңыз.

5. Сызықты кері айналдырыңыз

- Жолды кері қайтару үшін рекурсивті функцияны орындаңыз.

6. Ең үлкен ортақ бөлгішті табу (GCD)

- Екі санның gcd мәнін табу үшін рекурсивті функцияны жазыңыз.

7. Ханой мұнаралары

- Рекурсияның көмегімен Ханой мұнаралары мәселесін шешіңіз.

8. Палиндромды тексеру

- Жолдың палиндром екенін тексеру үшін рекурсивті функцияны орындаңыз.

9. Санның дәрежесін есептеу

- Санның дәрежесін есептейтін рекурсивті функцияны жаз.

10. Тізімдегі элементті табу

- Сұрыпталмаған тізімдегі элементті рекурсивті іздеуді жүзеге асыру.

11. Комбинацияларды есептеу

- \( n \) бастап \( k \) аралығындағы комбинациялар санын рекурсивті есептеу функциясын жазыңыз.

12. Рекурсивті сұрыптау

- Біріктірілген сұрыптау сияқты рекурсивті сұрыптау алгоритмін жүзеге асыру.

13. Графикті өту

- Графикті рекурсивті айналдыру функциясын жазыңыз (мысалы, тереңдік-бірінші іздеу).

def dfs(graph, n):  
 def dfs2(node, visited=None):  
 if visited is None:  
 visited = set()  
 visited.add(node)  
 print(node)  
 for neighbor in graph[node]:  
 if neighbor not in visited:  
 dfs2(neighbor, visited)  
 dfs2(n)  
  
  
graph = {  
 'A': ['B', 'C'],  
 'B': ['A', 'D', 'E'],  
 'C': ['A', 'F', 'G'],  
 'D': ['B'],  
 'E': ['B'],  
 'F': ['C'],  
 'G': ['C']  
}  
dfs(graph, 'A')

A

B

D

E

C

F

G

14. Лабиринттегі рекурсивті жолды табу

- Рекурсияның көмегімен лабиринттегі жолды табу функциясын орындаңыз.

15. Факториалды есте сақтау

- Есте сақтау арқылы рекурсивті факторлық есептеу функциясын оңтайландыру.

**Бағалау критерийлері:**

- Жеке есепті шешу үшін код жазу: 1 ұпай

- Қорғау кезінде жазылған кодты түсіндіру және түсіну: 2 ұпай

- Мұғалім таңдаған теориялық сұрақтардың біріне жауап: 1 ұпай

**Дайындық сұрақтары:**

1. Рекурсия дегеніміз не және ол бағдарламалау контекстінде қалай жұмыс істейді?

- Мақсаты: Оқушылардың рекурсия ұғымы туралы жалпы түсінігін тексеру.

2. Рекурсивті функцияның негізгі компоненттері қандай?

- Мақсаты: Студенттің рекурсивті функцияның құрылымын, оның ішінде негізгі жағдай мен рекурсивті қадамды түсінуін қамтамасыз ету.

3. Бағдарламалауда рекурсияны қолданудың артықшылықтары мен кемшіліктері қандай?

- Мақсаты: Оқушылардың рекурсияның артықшылықтарын (мысалы, кодтың қарапайымдылығы) және кемшіліктерін (мысалы, стектің толып кету қаупі) түсінуін бағалау.

4. Рекурсия ең жақсы шешім болып табылатын мәселеге мысал келтіріңіз.

- Мақсаты: Студент рекурсия тиімді тәсіл болатын сценарийлерді анықтай алатынын көру.

5. Рекурсияда қоңыраулар стекінің толып кетуін қалай болдырмауға болады?

- Мақсаты: Студент есте сақтау немесе итеративті тәсілге көшу сияқты терең рекурсиямен байланысты мәселелерді болдырмау әдістерін түсінеді.

6. Құйрық рекурсиясы дегеніміз не және ол не үшін маңызды?

- Мақсаты: Студенттердің құйрық рекурсиясы түсінігі және оның артықшылықтары туралы білімдерін тексеру.

7. Рекурсивті функцияда есте сақтау қалай жүзеге асырылады?

- Мақсаты: есте сақтау сияқты рекурсивті қоңырауларды оңтайландыру механизмдерін түсінуді бағалау.

8. Есептерді шешудің рекурсивті және итеративті тәсілдерін салыстыра аласыз ба?

- Мақсаты: Рекурсивті және итеративті әдістердің айырмашылығын түсіну және оларды салыстыра білу.

9. Рекурсивті функцияларды жазу кезінде қандай қателер болуы мүмкін және оларды қалай болдырмауға болады?

- Мақсаты: Студенттің негізгі регистрді жіберіп алу немесе дұрыс емес күй өзгерістері сияқты жиі кездесетін қателерді білуін қамтамасыз ету.

10. Ағаштар мен графиктер сияқты деректер құрылымдарын айналып өту үшін рекурсияны қалай пайдалануға болады?

- Мақсаты: Студенттердің күрделі деректер құрылымдарын өту үшін рекурсияны пайдалану туралы түсінігін тексеру.

**Примерная Задача для Лабораторной Работы 3**

**Задача: Рекурсивное Решение Задачи о Ханойских Башнях**

Задача о Ханойских башнях — классическая задача, решаемая с использованием рекурсии. Цель состоит в том, чтобы переместить диски с одного стержня на другой, придерживаясь определенных правил.

Правила:

1. За один раз можно перемещать только один диск.

2. Диск можно класть только на пустой стержень или на диск большего размера.

Цель задачи:

Написать рекурсивную функцию, которая выводит последовательность шагов для решения задачи о Ханойских башнях для n дисков.

Решение:

def hanoi\_towers(n, source, target, auxiliary):

"""

Решение задачи о Ханойских башнях.

n - количество дисков

source - начальный стержень

target - целевой стержень

auxiliary - вспомогательный стержень

"""

if n > 0:

# Переместить n-1 дисков на вспомогательный стержень

hanoi\_towers(n-1, source, auxiliary, target)

# Переместить оставшийся диск на целевой стержень

print(f"Переместить диск с {source} на {target}")

# Переместить n-1 дисков с вспомогательного на целевой стержень

hanoi\_towers(n-1, auxiliary, target, source)

# Пример использования функции

hanoi\_towers(3, 'A', 'C', 'B')

Объяснение:

Функция `hanoi\_towers` решает задачу о Ханойских башнях, используя рекурсивный подход.

- Если `n` больше 0, сначала перемещаем `n-1` дисков на вспомогательный стержень.

- Затем перемещаем самый большой диск на целевой стержень.

- После этого перемещаем `n-1` дисков с вспомогательного стержня на целевой, следуя тем же правилам.

Эта задача иллюстрирует классическое применение рекурсии и помогает студентам лучше понять концептуальные основы и практическую реализацию рекурсивных функций.