Логічне програмування, НаУКМА

Задачі на переливання

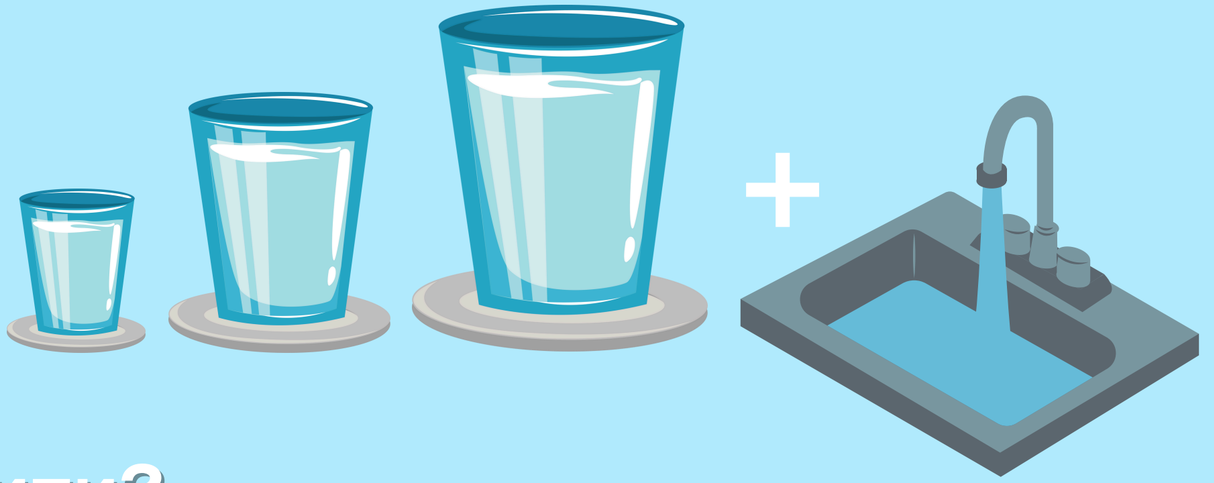
підготувала студентка 3 р.н.
спеціальності “Комп’ютерні науки”
Шулакова Катерина

Постановка задач

На вході:

1. **Набір посудин**, їхня максимальна ємність, поточна кількість води
2. **Ціль**: кількість води в одній з ємностей

1. **Набір посудин**, їхня максимальна ємність, поточна кількість води, **кран і злив**
2. **Ціль**: кількість води в одній з ємностей



Що робити?

Шляхом переливання води досягнути цілі.

! Переливання триває до повного спорожнення ємності-джерела або до повного заповнення приймаючої.

! правила для попередньої задачі + з джерела ємність наповнюється лише доповна, а в злив посудина спустошується повністю.

Як можна вирішити?

Ці задачі можна вирішувати на C/C++, Java, Python, Haskell тощо.... Але **зупинимося на Prolog**.

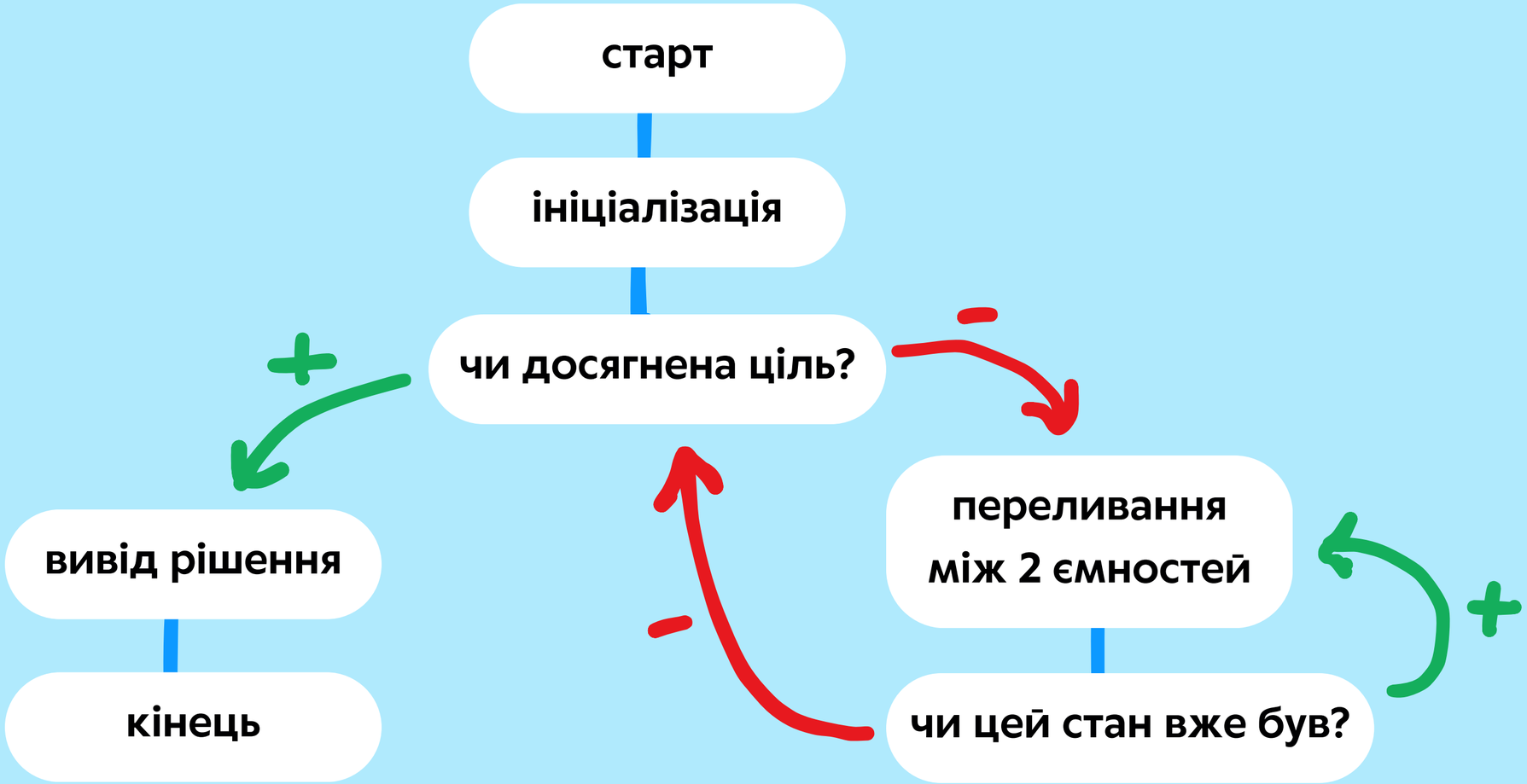
Чому пролог?

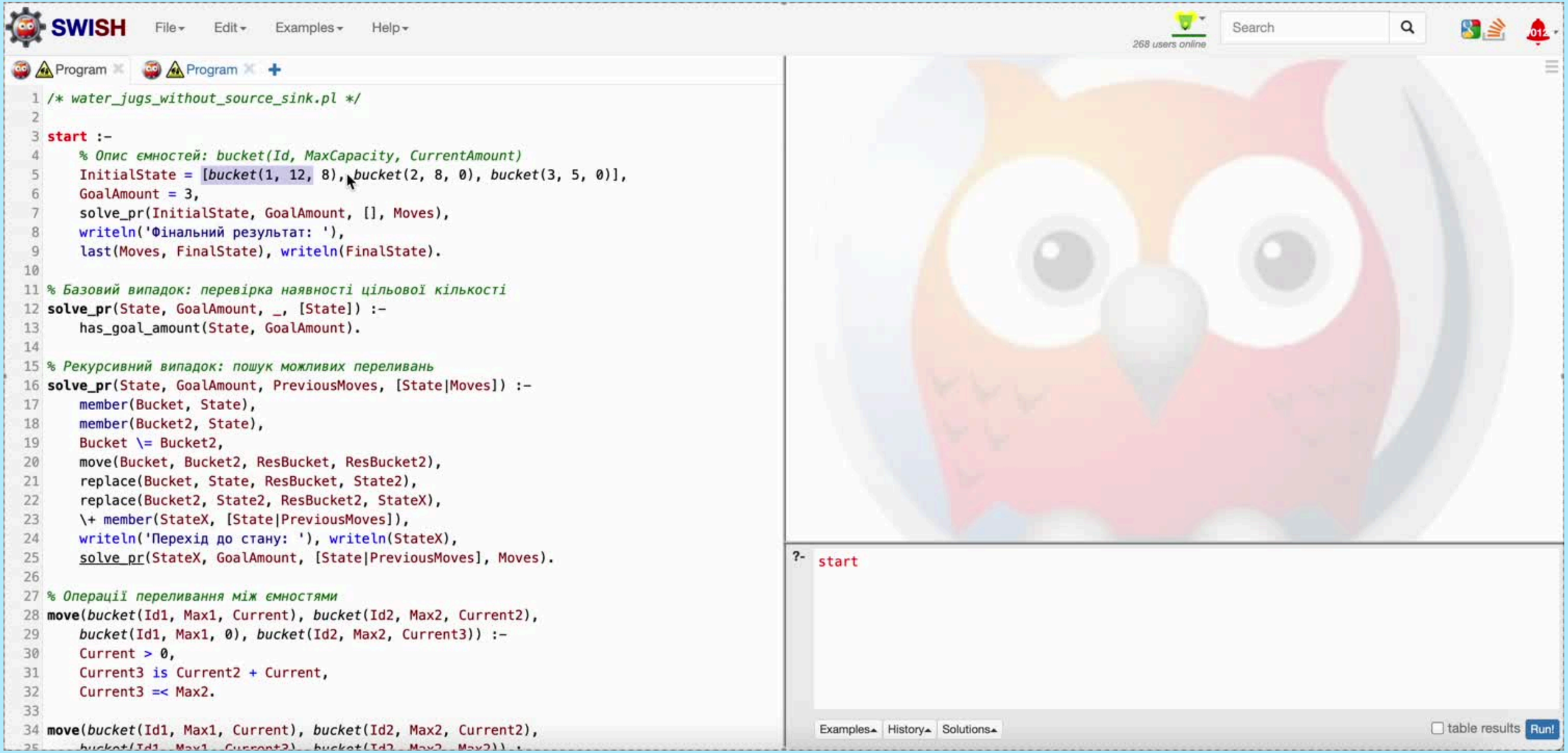
+ Природна підтримка логічного висновку та рекурсії.
+ Prolog заснований на декларативному підході, де описується логіка задачі через факти і правила.
+ Пошук з використанням backtracking.
+ Інтуїтивність опису задачі.

Незважаючи на всі переваги, варто враховувати деякі ризики:

ЧАС І РЕСУРСИ

Алгоритм


старт
ініціалізація
чи досягнена ціль?
+
-
вивід рішення
кінець
переливання
між 2 ємностей
чи цей стан вже був?
-
+


SWISH
File
Edit
Examples
Help
268 users online
Search
Program
Program
/* water_jugs_without_source_sink.pl */
start :-
% Опис ємностей: bucket(Id, MaxCapacity, CurrentAmount)
InitialState = [bucket(1, 12, 8), bucket(2, 8, 0), bucket(3, 5, 0)],
GoalAmount = 3,
solve_pr(InitialState, GoalAmount, [], Moves),
writeln('Фінальний результат: '),
last(Moves, FinalState), writeln(FinalState).
% Базовий випадок: перевірка наявності цільової кількості
solve_pr(State, GoalAmount, _, [State]) :-
has_goal_amount(State, GoalAmount).
% Рекурсивний випадок: пошук можливих переливань
solve_pr(State, GoalAmount, PreviousMoves, [State|Moves]) :-
member(Bucket, State),
member(Bucket2, State),
Bucket \= Bucket2,
move(Bucket, Bucket2, ResBucket, ResBucket2),
replace(Bucket, State, ResBucket, State2),
replace(Bucket2, State2, ResBucket2, StateX),
\+ member(StateX, [State|PreviousMoves]),
writeln('Перехід до стану: '), writeln(StateX),
solve_pr(StateX, GoalAmount, [State|PreviousMoves], Moves).
% Операції переливання між ємностями
move(bucket(Id1, Max1, Current), bucket(Id2, Max2, Current2),
bucket(Id1, Max1, 0), bucket(Id2, Max2, Current3)) :-
Current > 0,
Current3 is Current2 + Current,
Current3 =< Max2.
move(bucket(Id1, Max1, Current), bucket(Id2, Max2, Current2),
?-
start
Examples
History
Solutions
table results
Run!

Реальне застосування подібних задач

навчання

- Демонстрація алгоритмічних методів
- Логічне програмування

практика

- Розподіл ресурсів
- Автоматизація процесів
- Планування та логістика

Джерела

- GeeksforGeeks: Backtracking Algorithms - стаття, яка пояснює методи backtracking із прикладами застосування в алгоритмах.
- TutorialsPoint: Prolog Tutorial - посібник для початківців з Prolog, що містить пояснення основних принципів логічного програмування та приклади вирішення задач.
- “Задачі на переливання та зважування” курсу «Практикум з розв’язування олімпіадних та конкурсних задач» ст. викл. Рего В. Л.
- GeeksforGeeks: Water Jug Problem - детальний опис задачі з алгоритмічним підходом до її вирішення, включаючи псевдокод і приклади реалізації.