Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 18

Дисциплина	Функциональное и логическое программирование.
Тема	Рекурсия на Prolog.
Студент	Степанов А. О.
Группа	ИУ7-63Б
Оценка (баллы)	
Преподаватель	Толпинская Н.Б.

Москва, 2020 г.

ЗАДАНИЕ

Используя хвостовую рекурсию, разработать программу, позволяющую найти

1. n!,

2. n-е число Фибоначчи.

Листинг 1: Текст программы

```
1  predicates
2      factorial(integer, integer).
3      fibb(integer, integer).
4
5  clauses
6      factorial(0, Result) :- Result = 1, !.
7      factorial(N, Result) :-
8          NextN = N - 1,
9          factorial(NextN, NextResult),
10         Result = N * NextResult.
11
12     fibb(1, Result) :- Result = 1, !.
13     fibb(2, Result) :- Result = 1, !.
14     fibb(N, Result) :-
15         PN = N - 1, PPN = N - 2,
16         fibb(PN, PResult), fibb(PPN, PPResult),
17         Result = PResult + PPResult.
18
19 goal
20     write("5! : "),
21     factorial(5, Result);
22     write("f(10) : "),
23     fibb(10, Result).
```

РЕЗУЛЬТАТ РАБОТЫ

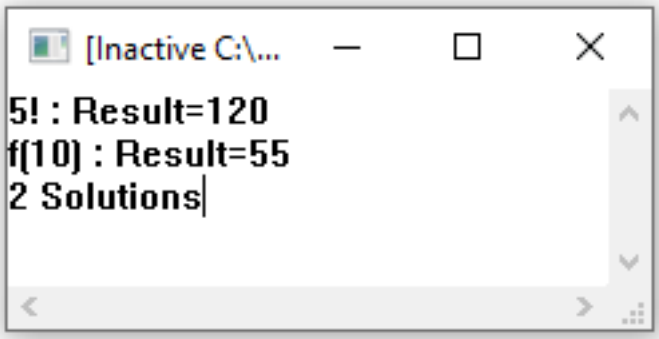

Рис. 1: Результат работы программы

ФОРМИРОВАНИЕ ОТВЕТА

Для одного из вариантов ВОПРОСА и каждого задания составить таблицу, отражающую конкретный порядок работы системы: Т.к. резольвента хранится в виде стека, то состояние резольвенты требуется отображать в столбик: вершина – сверху! Новый шаг надо начинать с нового состояния резольвенты!

Таблица 1: factorial(4, Result)

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат (и подстановка)	дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	factorial(4, Result)	Подстановка: N = 4, Result = Result	Прямой ход
2	NextN = N - 1 factorial(NextN, NextResult) Result=N*NextResult	Подстановка NextN = 3	Прямой ход
3	factorial(NextN, NextResult) Result=N*NextResult	Подстановка: N = 3, Result = NextResult	Прямой ход
4	NextN = N - 1 factorial(NextN, NextResult) Result=N*NextResult Result=N*NextResult	Подстановка NextN = 2	Прямой ход
5	factorial(NextN, NextResult) Result=N*NextResult Result=N*NextResult	Подстановка: N = 2, Result = NextResult	Прямой ход
6	NextN = N - 1 factorial(NextN, NextResult) Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка NextN = 1	Прямой ход
7	factorial(NextN, NextResult) Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка: N = 1, Result = NextResult	Прямой ход

8	NextN = N - 1 factorial(NextN, NextResult) Result=N*NextResult Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка NextN = 0	Прямой ход
9	factorial(NextN, NextResult) Result=N*NextResult Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка: N = 0, Result = NextResult	Прямой ход
10	Result = 1 Result=N*NextResult Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка: Result = 1	Прямой ход
11	Result=N*NextResult Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка: Result = 1	Прямой ход
12	Result=N*NextResult Result=N*NextResult Result=N*NextResult	Подстановка: Result = 2	Прямой ход
13	Result=N*NextResult Result=N*NextResult	Подстановка: Result = 6	Прямой ход
14	Result=N*NextResult	Подстановка: Result = 24	Прямой ход
15	Пусто	**Результат:** Result = 24	Обратный ход

Таблица 2: fibb(4, Result)

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат (и подстановка)	дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	fibb(4, Result)	Подстановка: N = 4, Result = Result	Прямой ход
2	PN = N - 1 PPN = N - 2 fibb(PN, PResult) fibb(PPN, PPResult)	Подстановка: PN = 3	Прямой ход

	Result = PResult + PPResult		
3	PPN = N - 2 fibb(PN, PResult) fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: PPN = 2	Прямой ход
4	fibb(PN, PResult) fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: N = 3, Result = PResult	Прямой ход
5	PN = N - 1 PPN = N - 2 fibb(PN, PResult) fibb(PPN, PPResult) Result = PResult + PPResult fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: PN = 2	Прямой ход
6	PPN = N - 2 fibb(PN, PResult) fibb(PPN, PPResult) Result = PResult + PPResult fibb(PPN, PPResult) Result = PResult + PPResult	PPN = 1	Прямой ход
7	fibb(PN, PResult) fibb(PPN, PPResult) Result = PResult + PPResult fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: N = 2, Result = PResult	Прямой ход
8	Result = 1 fibb(PPN, PPResult) Result = PResult + PPResult fibb(PPN, PPResult)	Подстановка: Result = 1	Прямой ход

	Result = PResult + PPResult		
9	fibb(PPN, PPResult) Result = PResult + PPResult fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: N = 1, Result = PPResult	Прямой ход
10	Result = 1 Result = PResult + PPResult fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: Result = 1	Прямой ход
11	Result = PResult + PPResult fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: Result = 2	Прямой ход
12	fibb(PPN, PPResult) Result = PResult + PPResult	Подстановка: N = 2	Прямой ход
13	Result = 1 Result = PResult + PPResult	Подстановка: Result = 1	Прямой ход
14	Result = PResult + PPResult	Подстановка: Result = 3	Прямой ход
15	Пусто	**Результат:** Result = 3	Обратный ход

ВОПРОСЫ

1. **Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как организовать выход из рекурсии в Prolog?**

   Рекурсия — это ссылка на определяемый объект во время его определения. В языке Prolog рекурсия организуется при помощи правила, в котором есть обращение к тому же правилу. Выход из рекурсии в Prolog организуется при помощи отсечения.

2. **Какое первое состояние резольвенты?**

   Пролог выполняет унификацию в двух случаях:

   — когда цель сопоставляется с заголовком предложения;

   — когда используется знак равенства, который является инфиксным предикатом (предикатом, который расположен между своими аргументами, а не перед ними).

   Первое состояние резольвенты – вопрос.

3. **В каком случае система запускает алгоритм унификации? Каково назначение использования алгоритма унификации? Каков результат работы алгоритма унификации?**

```
goal
    P1 = birthday(person("Ivan", "Petrov"), date("August", 2, 1980)),
    P1 = birthday(Name, date(_, _, 1980)), write(Name).
```

   При согласовании первой подцели переменная P1 получит значение, указанное справа от знака “=”. При согласовании второй подцели P1 уже связана. Так как термы, находящиеся по обе стороны знака “=” сопоставимы, то переменная Name будет связана со значением person(“Ivan”, “Petrov”). При согласовании третьей подцели, стандартного предиката write, будет напечатано значение связанной переменной Name.

4. **В каких пределах программы переменные уникальны?**

   Областью действия переменной в Prolog является одно предложение. В разных предложениях может использоваться одно имя перменной для обозначения разных объектов. Исключением является анонимная переменная. Каждая анонимная переменная – это отдельный объект.

5. **Как применяется подстановка, полученная с помощью алгоритма унификации?**

   Если унификация прошла успешно, то применяется подстановка. Переменные связываются со значениями.

6. **Как изменяется резольвента?**

   Преобразование резольвенты выполняется с помощью редукции.

   Редукция – замена цели телом того правила, заголовок которого унифицируется с целью. Новая резольвента получается в два этапа:

7. **В каких случаях запускается механизм отката?**

   В том месте программы, где возможен выбор нескольких вариантов, Пролог сохраняет в специальный стек точку возврата для последующего возвращения в эту позицию. Точка возврата содержит информацию, необходимую для возобновления процедуры при откате. Выбирается один из возможных вариантов, после чего продолжается выполнение программы.

   Во всех точках программы, где существуют альтернативы, в стек заносятся указатели. Если впоследствии окажется, что выбранный вариант не приводит к успеху, то осуществляется откат к последней из имеющихся в стеке точек программы, где был выбран один из альтернативных вариантов. Выбирается очередной вариант, программа продолжает свою работу. Если все варианты в точке уже были использованы, то регистрируется неудачное завершение и осуществляется переход на предыдущую точку возврата, если такая есть. При откате все связанные переменные, которые были означены после этой точки, опять освобождаются.