I. Ներածություն

Prolog-ը (***Pro***graming in ***Log***ic) տրամաբանական ծրագրավորման լեզու է: Prolog լեզվով ծրագրավորելիս խնդրի լուծումը նախ ներկայացվում է մուտքային և ելքային տվյալները միմյանց հետ կապող նպատակային պրեդիկատի միջոցով, այնուհետև խնդիր է դրվում կառուցելու փաստերից և արտածման կանոններից բաղկացած տրամաբանական համակարգ, որից ֆորմալ եղանակով հնարավոր լինի արտածել նպատակային պրեդիկատը:

Prolog լեզվում բացակայում են պրոցեդուրային լեզուներին (C++, Java, C# և այլն) բնորոշ տվյալների տիպերը և գործողությունների հաջորդականության ղեկավարումը: Փոխարենը օգտագործվում են պրեդիկատների սահմանման անդրադարձ (ռեկուրսիվ) կանոններ, ինչի շնորհիվ Prolog-ը վերածվում է դեկլարատիվ լեզվի: Այն ծրագրավորողին թելադրում է անդրադարձ մտածելակերպ և անդրադարձ ոճով խնդիրների լուծում: Prolog լեզուն հարմար է օգտագործել այնպիսի կիրառություններում, ինչպիսիք են *անդրադարձ տվյալների կառուցվածքների մշակումը*, *սիմվոլային հաշվումները*, *փորձագիտական համակարգերի ստեղծումը*, *թեորեմների ապացույցը*, *տեքստերի թարգմանությունը*:

1.1. Փաստեր, կանոններ, հարցումներ

1.1.1. Փաստեր

Փաստը Prolog լեզվում գրառվում է հետևյալ կերպ.

հարաբերություն (օբյեկտների ցուցակ).

որտեղ հարաբերությունը և օբյեկտները փոքրատառով սկսվող անուններ են: Նշենք, որ փաստը պարտադիր պետք է ավարտվի «կետ» նշանով: Օրինակ՝

likes(john, mary).

փաստով արձանագրվում է այն, որ **john**-ը հավանում է **mary**-ին: Եթե անհրաժեշտ է նշել, որ **mary**-ն նույնպես հավանում է **john**-ին, ապա պետք է նաև ավելացնել հետևյալ փաստը.

likes(mary, john).

Փաստերի հաջորդականությունը Prolog լեզվում կոչվում է պարզագույն տվյալների բազա: Օրինակ՝

```
likes(joe, fish).
likes(joe, mary).
likes(mary, book).
likes(joe, book).
```

1.1.2. Հարցումներ

Տվյալների բազային կարելի է դիմել հարցումներով: Հարցումը գրառվում է այնպես, ինչպես փաստն այն տարբերությամբ, որ հարցումից առաջ դրվում են **?-** նշանները: Օրինակ՝

?- likes(joe, money).
no

?- likes(mary, joe).
no

?- likes(mary, book).
yes

1.1.3. Փոփոխականներ

Դիցուք անհրաժեշտ է պարզել, թե ի՞նչ է հավանում **joe**-ն: Դա կարելի էր անել հաջորդաբար հարցնելով՝ հավանում է արդյոք **joe**-ն այս կամ այն օբյեկտը: Փոխարենը կարելի է կազմել հետևյալ հարցումը.

?- likes(joe, X).

Այստեղ **X**-ը հանդես է գալիս փոփոխականի դերում, ինչը բխում է գրելաձևից. Prolog լեզվում մեծատառով սկսվող ցանկացած անուն դիտարկվում է որպես փոփոխական: Փոփոխականները կարող են լինել արժևորված կամ անորոշ: Նշենք, որ Prolog լեզվում փոփոխականներն ըստ տիպերի չեն դասակարգվում և ցանկացած օբյեկտ կարող է լինել ցանկացած փոփոխականի արժեք: Կասենք, որ փոփոխականը կոնկրետացված է, եթե տվյալ պահին նրան համապատասխանում է որևէ օբյեկտ: Այլապես կասենք, որ այն կոնկրետացված չէ:

Եթե հարցումը պարունակում է փոփոխականներ, ապա Prolog համակարգը հաջորդաբար դիտարկում է տվյալների բազայում գրառված փաստերը՝ որոնելով փոփոխականների այնպիսի կոնկրետացում, որի դեպքում հարցումը համընկնում է որևէ փաստի հետ: Ընդհանուր դեպքում, երբ տվյալների բազան պարունակում է կանոններ, փոփոխականների կոնկրետացումը կատարվում է հետդարձով որոնման եղանակով (**backtracking**):

Վերը դիտարկված օրինակի համար **X**-ն առաջին անգամ կոնկրետանում է՝ ընդունելով **fish** արժեքը: Դրանից հետո համակարգը սպասում է հետագա գործողությունների: Եթե հրահանգվում է շարունակել որոնումը, ապա **X**-ը կոնկրետանում է երկրորդ անգամ՝ ընդունելով **mary** արժեքը: Որոնումը կրկին շարունակելու դեպքում **X**-ը կոնկրետանում է երրորդ և վերջին անգամ՝ ընդունելով **book** արժեքը: Այսպիսով՝ ստանում ենք.

?- likes(joe, X).
X = fish;
X = mary;
X = book;
yes

1.1.4. Կոնյունկցիաներ

Դիտարկենք տվյալների հետևյալ բազան:

```
likes(mary, food).
likes(mary, wine).
likes(john, wine).
likes(john, mary).
```

Դիցուք անհրաժեշտ է պարզել՝ հավանում են արդյոք **john**-ը և **mary**-ն միմյանց, թե ոչ: Դա կարելի էր անել՝ նախ կատարելով.

?- likes(john, mary).

հարցումը և այնուհետև դրական պատասխան ստանալուց հետո՝

?- likes(mary, john).

հարցումը: Փոխարենը բավարար է կատարել հետևյալ համակցված հարցումը.

?- likes(john, mary), likes(mary, john).
no

Օգտագործելով փոփոխականներ կոնյունկցիաներում՝ կարող ենք կառուցել ավելի բարդ հարցումներ: Օրինակ՝

?- likes(john, X), likes(mary, X).
X = wine;
yes

1.1.5. Կանոններ

Կանոնները նշվում են հետևյալ կերպ.

վերնագիր:- մարմին.

(«**:-**» նշանակումը կարդացվում է «եթե»):

Օրինակ՝

likes(john, X):- likes(X, wine), likes(X, food).

կամ՝

likes(john, X):- woman(X), likes(X, wine).

Նշենք, որ կանոններում կարող են օգտագործվել ցանկացած թվով փոփոխականներ:

Ստորև բերված է տվյալների բազա, որը պարունակում է Անգլիայի Վիկտորյա թագուհու տոհմածառին առնչվող որոշ փաստեր.

```
man(albert).
man(edward).
woman(alice).
woman(victoria).
parents(edward, victoria, albert).
parents(alice, victoria, albert).
```

Սահմանենք **isSister** հարաբերությունը հետևյալ կանոնի միջոցով.

```
isSister(X, Y):-  woman(X), parents(X, M, F), parents(Y, M, F).
```

Ավելացնենք այս կանոնը տվյալների բազային և ձևակերպենք հետևյալ հարցումը.

```
?- isSister(alice, X).
X = edward;
X = alice;
yes
```

X = alice անհեթեթ պատասխանը բացառելու համար **isSister** կանոնի աջ մասում պետք է պահանջել, որ **X**-ը և **Y**-ը հավասար չլինեն միմյանց:

Դիտարկենք մեկ այլ օրինակ.

```
thief(john).
likes(mary, food).
likes(mary, wine).
likes(john, X):- likes(X,wine).
can_steal(X, Y):- thief(X), likes(X, Y).
```

Նշենք, որ այս օրինակում **likes** պրեդիկատը օգտագործվում է միաժամանակ և՛ փաստերում, և՛ կանոններում:

```
?- can_steal(john, X).
X = mary;
yes
```

Վարժություն

Տրված են հետևյալ հարաբերությունները.

father(X, Y)	(**X**-ը **Y**-ի հայրն է)
mother(X, Y)	(**X**-ը **Y**-ի մայրն է)
man(X)	(**X**-ը տղամարդ է)
woman(X)	(**X**-ը կին է)
parent(X, Y)	(**X**-ը **Y**-ի ծնողն է)
different(X, Y)	(**X**-ը և Y-ը տարբեր են)

Հիմք ընդունելով այս հարաբերությունները՝ սահմանել հետևյալ նոր հարաբերությունները.

1. **isMother(X)** (**X**-ը մայր է)
2. **isSon(X)** (**X**-ը որդի է)
3. **sister(X, Y)** (**X**-ը **Y**-ի քույրն է)
4. **grandfather(X, Y)** (**X**-ը **Y**-ի պապն է)