پروژه شماره ۴

# برنامه تولید سری فیبوناچی

## به همراه بررسی امکان تولید Nامین جلمه سری

## با برنامه نویسی پرولوگ

استاد: جناب آقای دکتر فیضی درخشی

دانشجو: پارسا یوسفی نژاد محمدی

شماره دانشجویی: ۱۴۰۰۵۳۶۱۱۰۴۸

# هدف پروژه

**در این پروژه قصد داریم برنامه‌ای به زبان پرولوگ بنویسیم که Nامین عدد از سری فیبوناچی را محاسبه کند و یا اینکه بررسی کند که عدد N، nامین عدد از سری با مقدار F است یا نه، همچنین از برنامه انتظار داریم که N عدد اول سری فیبوناچی را در صورت امکان تولید و پرینت کند. برای این مساله همچون پروژه قبل، یک رابطه ریاضی غیرمستقیم به شکل بازگشتی وجود دارد که به کمک آن سعی در طراحی و تشریح این برنامه خواهیم کرد.**

# تشریح پروژه

این برنامه دارای دو قانون است که اولین قانون fib دو پارامتر N و F را می‌گیرید و بررسی می‌کند که آیا N، nامین عدد سری فیبوناچی با مقدار F است یا خیر، و دیگری قانون producefib است که تنها یک پارامتر به نام N می‌گیرید و سعی می‌کند که N جمله اول دنباله را تولید بکند.

الگوریتم و شیوه پیاده سازی قانون fib برگرفته از رابطه بازگشتی ریاضی، برای محاسبه اعداد دنباله فیبوناچی است:

$$fib(N) = fib(N-1) + fib(N-2)$$
$$fib(1) = 0 \qquad fib(2) = 1$$

همانطور که مشخص است، برای محاسبه Nامین عدد از سری فیبوناچی باید دو جمله قبلی آن را محاسبه کنیم و سپس آن دو جمله قبلی را با هم جمع و ادغام کنیم تا نتیجه بدست آید، همچنین لازم است که این روند بازگشتی برای تمام جملات قبل‌تر تا نرسیدن به شرطهای توقف اجرا شود، و پس رسیدن به شروط پایانی که از مساله می‌دانیم، جایگذاری پسرو را پیش بگیریم و مقادیر جملات قبل‌تر را بدست آورده و با جایگذاری‌های پی‌درپی بتوانیم مقدار خواسته شده را محاسبه کنیم، مرتبه زمانی بدترین حالت (worst case) این الگوریتم بازگشتی به سادگی با استفاده از معادله مشخصه همگن قابل محاسبه بوده:

$$\text{Time Complexity} : O(2^n)$$

در این الگوریتم بدلیل اینکه نتایج هربار فراخوانی ذخیره نمی‌شود، (Divide & Conqure) محاسبات تکراری بسیار زیادی در حال انجام است، که موجب افزایش پیچیدگی زمانی برنامه می‌گردد.

برای جلوگیری از این اتفاق، شیوه برنامه ریزی پویا (Dynamic Programming) توصیه می‌شود تا از این محاسبات تکراری جلوگیری شود.

حال به بررسی نحوه پیاده سازی قوانین و واقعیت‌های برنامه میپردازیم، دانسته‌ها و شرایط توقف برنامه را fact تعریف میکنیم:

```
fib(1,0).    fib(2,1).
```

این دو واقعیت به این دانسته‌ها اشاره دارند که اولین جمله از دنباله فیبوناچی عدد ۰ و دومین جمله از آن، مقدار ۱ می‌باشد.

حال به پیاده سازی قانونی می‌پردازیم که بررسی می‌کند، آیا Nام عدد از دنباله با مقدار F تطابق دارد یا خیر، بدین منظور از رابطه بازگشتی ریاضی بهره میگیریم:

```
1 % calculates Nth-Fibonacci number which must equals to F to return true
2 fib(N,F):-
3     N⩾3,
4     N1 is N-1,
5     N2 is N-2,
6     fib(N1,F1),
7     fib(N2,F2),
8     F is F1 + F2.
```

در ابتدا بررسی میکنیم که آیا N مقدار مجازی دارد یا نه، این کار را با شرط N>=3 بررسی می‌کنیم، به این دلیل عدد ۳ را انتخاب کردیم، چرا که در واقعیت‌هایمان، اولین و دومین اعداد دنباله را داریم، و از آن به بعد نیاز به استفاده از این قانون پیدا می‌کنیم، سپس طبق رابطه بازگشتی بالا، باید قانون fib را برای دو جمله قبلی محاسبه کنیم، که این کار را با دستورات سطرهای ۴ تا ۷، که متناسب با کامپایلر swi-prolog هستند انجام دادیم. یعنی باید F1 و F2 ای برای درست بودن این قانون طوری پیدا شوند که جمع آن‌ها همان مقدار اصلی F را تولید بکند، اگر چنین Fهایی پیدا شوند، آنگاه کامپایلر به کوئری ما پاسخ .true را برمی‌گرداند و در غیر این صورت که F نمی‌تواند از حاصل جمع‌های Fهای جملات قبل‌تر از خود نتیجه شود، پاسخ .false را return میکند که این موضوع در سطر ۸ام کد

بررسی شده است، در ادامه شیوه‌های کوئری دادن به این قانون، مورد بررسی قرار خواهد گرفت.
حال به پیاده سازی، قانون producefib میپردازیم که تنها یک پارامتر N را دریافت می‌کند و N جمله اول سری فیبوناچی را با کمک متد بازگشتی تولید و چاپ می‌کند:

```
1  % produces first N number of Fibonacci series, and then prints them
2  producefib(N):-
3      N⩾1,
4      fib(N,Answer),
5      M is N-1,
6      format('~w ',[Answer]),
7      producefib(M).
```

در کد فوق، می‌بینیم که N بار قانون fib فراخوانی می‌شود، از طرفی، مرتبه اجرایی آن را از پیش $\mathbf{O(2^N)}$ بدست آوریدم، پس به سادگی می‌توان نتیجه گرفت که مرتبه اجرایی این قانون، برابر با $\mathbf{O(N\times 2^N)}$ خواهد بود.
وظیفه این قسمت از برنامه، این است که قانون دو پارامتره fib(N,F) را به ازای N های مختلف از عدد N تا عدد ۱ فراخوانی کرده و پاسخ Answer تولید شده توسط پرولوگ را با کمک متد format که Answer را به عنوان تنها عنصر از لیست آرگومان های چاپی دریافت می‌کند، در ترمینال چاپ کند، این برنامه به کمک فراخوانی fib(N,Answer) سعی در یافتن پاسخ Nامین عدد حال حاضر در دنباله فیبوناچی می‌کند و بعد از چاپ، مقدار N را یک واحد کاهش و سپس به فراخوانی بازگشتی خودش می‌پردازد و همانطور که مشخص است، شرط توقف زمانی اتفاق می‌افتد که N به مقداری کمتر از ۱ برسد که در این حالت Nامین عدد دنباله معنی ندارد و شرط نقض میشود و با false. پاسخ کوئری ما به اتمام می‌رسد.

## ❖ کوئری دادن به برنامه:

### ▪ بررسی صحت کوئری

در این قسمت پرسشی را جهت بررسی صحت رابطه‌امان از پرولوگ می‌پرسیم، یعنی اینکه می‌خواهیم بدانیم که آیا رابطه fib برای N و F دلخواهی برقرار است یا نه، در صورت برقرار بودن،

پرولوگ پیام .true را برمی‌گرداند که نشان دهنده این موضوع است که عدد F، Nامین عدد از دنباله فیبوناچی بوده است:

```
N =: 1, 2, 3, 4, 5, 6, 7,  8,  9, 10, 11, 12,  13, ...
F =: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
```

```
?- fib(5,3).
true .

?- fib(8,13).
true .

?- fib(13,83).
false.
```

```
?- fib(13,94).
false.

?- fib(12,89).
true .

?- fib(1,0).
true .
```

- **یافتن پاسخ Nامین عدد دنباله فیبوناچی**

نوع دیگر کوئری دادن به پرولوگ، درخواست یافتن Nامین عدد از دنباله می‌باشد، بدین منظور باید سوالمان را اینگونه مطرح کنیم که پارامتر مجهول F را متغیری دلخواه چون X بگذاریم تا پرولوگ چنین Xای را که متناسب با شرایط درستی کوئری است را پیدا کند و نتیجه را برایمان، که همان Nامین عدد دنباله است را بازگرداند:

```
?- fib(8,X).
X = 13 .

?- fib(17,X).
X = 987 .

?- fib(1,X).
X = 0 .
```

```
?- fib(-10,X).
false.

?- fib(3,X).
X = 1 .

?- fib(0,X).
false.
```

- **درخواست چاپ N عدد اول دنباله فیبوناچی**

این کوئری از قانون producefib(N) استفاده می‌کند، همانطور که در بخش بررسی کد producefib(N) دیدیم، با کمک این قانون و دادن این کوئری با مقدار دلخواه و معتبر N، شاهد چاپ شدن N عدد اول از دنباله در ترمینال پرولوگ خواهیم بود:

*توجه: علت .false برگرداندن در آخر هر کوئری صحیحی، بدلیل نقض شدن شرط N>=1 در آخرین فراخوانی بازگشتی است که منجر به قطع شدن روند فراخوانی‌های متعدد بازگشتی و به اتمام رساندن صحیح کوئری می‌شود.

```
?- producefib(4).
2 1 1 0
false.

?- producefib(8).
13 8 5 3 2 1 1 0
false.

?- producefib(11).
55 34 21 13 8 5 3 2 1 1 0
false.
```

```
?- producefib(-3).
false.

?- producefib(-9).
false.

?- producefib(0).
false.
```

Parsa Yousefi Nejad

# جمعبندی پروژه

**در این پروژه توانستیم روند صفر تا صد طراحی یک برنامه کاربردی پرولوگ را برای دنباله فیبوناچی را تشریح کنیم، در ابتدا به بررسی رابطه ریاضی فیبوناچی و سپس مرتبه زمانی آن الگوریتم پرداختیم، سپس کد مساله را قطعه قطعه مدل‌سازی و پیاده کردیم و بعد از آن به شیوه های مختلف کوئری دادن به برنامه پرداختیم، این برنامه میتواند، سه خواسته کاربر را برآورده کند:۱- یافتن Nایم عدد دنباله ۲- چاپ Nعدد اول دنباله ۳- بررسی وجود رابطه fib بین N و F.**

**همچنین در این پروژه از فراخوانی‌های بازشگتی قوانین استفاده کردیم و توانستیم مهارت‌های برنامه نویسی پرولوگ را با انجام این پروژه در خودمان بهبود بدهیم.**