راهنمای هوش مصنوعی

با ویژوال پرولوگ

مولف:

دکتر محمد علی ترکمانی

سرشناسه	:	ترکمانی، محمدعلی، ۱۳۵۴ -
عنوان و نام پدیدآور	:	راهنمای هوش مصنوعی با ویژوال پرولوگ [کتاب]/مولف محمد علی ترکمانی.
مشخصات نشر	:	مشهد: ارسطو ، ۱۳۹۴.
مشخصات ظاهری	:	۲۶۰ ص.: مصور
شابک	:	978-600-7940-93-8
وضعیت فهرست نویسی	:	فیپا
موضوع	:	هوش مصنوعی -- داده‌پردازی
موضوع	:	پرولوگ (زبان برنامه‌نویسی کامپیوتر)
رده بندی کنگره	:	Q۳۳۶/ت۴ر۲ ۱۳۹۴
رده بندی دیویی	:	۰۰۶/۳۰۲۸۵
شماره کتابشناسی ملی	:	۴۱۱۱۳۳۷

نام کتاب : راهنمای هوش مصنوعی با ویژوال پرولوگ

موضوع: ویژوال پرولوگ

موضوع: هوش مصنوعی

مولف : دکتر محمد علی ترکمانی

ناشر : ارسطو (با همکاری سامانه اطلاع رسانی چاپ و نشر ایران)

صفحه آرایی ، تنظیم و طرح جلد : علی بیات

تیراژ : ۱۰۰۰ جلد

نوبت چاپ : دوم – ۱۳۹۸

تعداد صفحات: ۲۵۴ ص

چاپ : مدیران

قیمت : ۵۰۰۰۰ تومان

تلفن های مرکز پخش : ۰۹۱۷۷۱۶۴۹۴۰ – ۵۰۹۶۱۴۶ - ۰۵۱۱

وب سایت: **www.chaponashr.ir/Torkamani**

فهرست مطالب

مقدمه:

پرولوگ یک زبان برنامه نویسی است که در ابتدا برای تولید برنامه های منطقی و توصیفی و همچنین سیستمهای هوشمند طراحی شد. بعد از عرضه ویندوز XP به بازار، نسخه ویژوال این زبان نیز وارد بازار گردید و به شدت مورد توجه متخصصان قرار گرفت. Visual prolog تقریباً تمامی امکانات یک زبان برنامه نویسی ویژوال را دراختیار برنامه نویس قرار می‌دهد و در کنار آن امکانات مناسبی برای برنامه نویسی منطقی و توصیفی فراهم نموده است. امروزه از Visual prologبیشتر برای توسعه سیستمهاهای هوشمند و سیستم‌های خبره استفاده می‌شود.

هدف از این کتاب یادگیری زبان ویژوال پرولوگ و محیط توسعه آن به منظور پیاده سازی مسائل هوش مصنوعی و توسعه برنامه‌های آن است. در این کتاب ساختار زبان Visual prolog نظیر انواع داده‌ها، عملگرها، فرم‌ها، رویدادهای ماوس، Closureها، کلاس‌ها و اشیاء، برنامه‌های کنسول، لیست‌ها، عملیات رشته‌ای، اشکال زدایی برنامه‌ها و ... بررسی خواهد شد. همچنین تعداد زیادی از تکنیک‌های هوش مصنوعی نظیر الگوریتم Min-Max، جستجوی عمقی، جستجوی سطحی، جستجوی Heuristic ، هرس آلفا-بتا، بررسی شده و پیاده سازی مسائل کلاسیک و نوین هوش مصنوعی و سیستم های خبره نظیر بازی kala، مسئله n وزیر، مسئله "کشاورز، گرگ، بز و کلم"، برنامه تشخیص بیماری، مسئله maze، مسئله برجهای هانوی، Sudoku Puzzle و شبکه های عصبی با استفاده از آخرین نسخه Visual prolog شرح داده می‌شود. با توجه به عدم وجود منابع فارسی در این حوزه، این کتاب می‌تواند برای اساتید، دانشجویان و متخصصان هوش مصنوعی مفید باشد. امید است این اثر مورد توجه همکاران و دانشجویان گرامی قرار گیرد. از اساتید و دانشجویان گرامی تقاضا دارم دیدگاه‌های خود را از طریق ایمیل m.a.torkamani@gmail.com با اینجانب در میان بگذارند تا انشالله در ویرایش‌های بعدی اشکالات یا کاستی‌های احتمالی کتاب مورد تجدید نظر قرار گیرد. در پایان وظیفه خود می‌دانم از زحمات آقای مهندس علی بیات به خاطر طراحی جلد کتاب و همچنین از مدیریت انتشارات ارسطو و سامانه اطلاع‌رسانی چاپ و نشر ایران، جناب آقای حسین قنبری، تشکر و قدردانی نمایم.

محمد علی ترکمانی

آبان ۹۸

فصل اول

معرفی و نصب ویژوال پرولوگ

۱-۱-پرولوگ چیست؟

زبان برنامه نویسی پرولوگ[1] در سال ۱۹۷۲ در دانشگاه مارسی توسط آلین کولمِرِر و همکارانش ابداع شد. Prolog مخف عبارت فرانسوی "Programmation en Logique" است. هدف اصلی طراحی این زبان برنامه نویسی کمک به ساختن سیستمی بود که بتواند چنین کاری را انجام بدهد:

کاربر:

- گربه‌ها موش را می‌کشند.
- Tom یک گربه است که موش را که پنیر می‌خورد دوست ندارد.
- Jerry یک موش است که پنیر می‌خورد.
- Max موش نیست.
- Tom چه کاری انجام می‌دهد؟

کامپیوتر

- Tom موش را که پنیر می‌خورد دوست ندارد.
- Tom موش را می‌کشد.

کاربر

- Jerry چه چیزی می‌خورد؟

---

[1] Prolog

کامپیوتر

- پنیر.

پرولوگ یک زبان سطح بالا است و قدرت آن در استدلال و استنتاج است و از آن در زمینه‌های زیر استفاده شده است:

- سیستم‌های خبره
- درک زبان‌های طبیعی
- اثبات نظیریه
- طراحی ابزار جهت طراحی کامپیوتری
- نوشتن کامپایلر
- برنامه نویسی سیستمهای هوشمند و هوش مصنوعی

پرولوگ بر است دو بنیان ریاضی بنا شده است:

۱. یکسان‌سازی (تطبیق الگو)

۲. جستجو (عقبگرد)

۲-۱-تفاوت پرولوگ و زبانهای رویه‌ای

در زبانهای رویه‌ای[1] با توابع و زیر برنامه ها کار می‌کنیم. در اینگونه زبان‌ها فراخوانی توابع و زیربرنامه ها با استفاده از نام آنها و ارسال پارامترهای ورودی و خروجی انجام می‌شود. توابع و زیربرنامه ها هر دو می‌توانند چند پارامتر ورودی را دریافت کنند، اما یک تابع فقط می‌تواند یک مقدار را برگرداند ولی زیر برنامه می‌توانند بیش از یک مقدار را برگرداند.

پرولوگ هم شباهت‌هایی با زبانهای رویه‌ای دارد. زبانهای رویه‌ای با Procedure ها کار می کنند، اما پرولوگ با Predicate ها کار می‌کند. از Predicate ها برای ایجاد یا نوشتن قوانین (Rules) استفاده می‌شود. اگر Rule دارای داده‌های استاتیک باشد، به یک Fact اشاره دارد. از سوی دیگر اگر از متغیرها و واژه if استفاده کنید، حتماً باید Predicate ها را در یک بخش که clauses

[1] Procedural

نامیده می‌شود بنویسید. بنابراین بخش Predicate ، Predicate ها و پارامترهای آنها را لیست می‌کند در حالی که بخش clauses ، Predicate ها را تعریف می‌کند.

مثال:

```
predicates
CallingName(parometers)
name(Name string).
isParent( Parent string, Child string)
Clauses
name ("Dick").  % This is a Fact
name("Sally").  % This is a Fact
name("John").  % This is a Fact
name("Larry").  % This is a Fact
isParent("John", "Sally").  % This is a Fact
isParent("John", "Larry").  % This is a Fact
isParent(P, "Dick") if name(P) and isParent(P, "Larry"). %This is a Rule"
```

یک بخش جالب در پرولوگ Rule construction tools است که از سه اپراتور تشکیل شده است که عبارتند از کلمه کلیدی if و اپراتورهای and و or (خط آخر برنامه فوق را ملاحظه کنید).

صرف نظر از کاماها و خطوط خالی، برنامه‌های پرولوگ دنباله از کِلازها می‌باشند. کِلازها یا حقایق‌اند یا قواعد.

نکته: در بخش Clauses، انتهای هر RULE یا Fact با یک نقطه مشخص می‌شود.

مثال دیگر:

```
Clauses
dog("Rex").
cat("Tom").
animal(X):-dog(X).
```

dog(“Rex”) و cat(“Tom”) مثال‌هایی از حقایق هستند. اگر بخواهیم آن‌ها را به زبان طبیعی و فارسی بیان کنیم، می‌گوئیم: "Rex یک سگ است" و" Tom یک گربه است".

dog یک محمول(predicate) نامیده می‌شود. این محمول یک آرگومان به نام Rex دارد که درون () قرار گرفته است.

آخرین خط برنامه یعنی .animal(X):-dog(X) یک قاعده نام دارد. علامت- : (نقل قول و خط فاصله) می‌تواند به عنوان if تلقی گردیده و خوانده شود. X یک متغیر است.. در این متن X بیانگر هر مقداری می‌باشد.

قاعده ارائه شده در مثال فوق می‌توانند به صورت زیر خوانده شود:

X حیوان است اگر X یک سگ باشد (برای هر x)

استنباط اینکه Rex یک حیوان است با توجه به کلازهای فوق، برای هر کسی به راحتی میسر است. پرولوگ نیز می‌تواند چنین نتیجه‌ای استنباط کند. با این وجود هیچ سندی دال بر اینکه filix نیز یک حیوان است وجود ندارد و بنابراین حاصل .animal("Rex") غلط یا No خواهد بود.

مثال:

```
female("Maryam").  % This is a Fact
parent(Naser,X) :- parent(nancy,X).
```

خط اول یک Fact و خط دوم نمونه ای از یک قانون است.

این قانون می گوید که Naser واللدین (parent) یک شخص است اگرMaryam یک شخص (person) باشد.

۱-۳-ویژوال پرولوگ چیست؟

بعد از عرضه ویندوز XP به بازار، نسخه ویژوال زبان برنامه نویسی پرولوگ نیز وارد بازار گردید و به شدت مورد توجه متخصصان قرار گرفت. Visual prolog تقریباً تمامی امکانات یک زبان برنامه نویسی ویژوال را دراختیار برنامه نویس قرار می‌دهد و در کنار آن نیز امکانات مناسبی برای برنامه نویسی منطقی و توصیفی فراهم نموده است. امروزه از Visual prolog بیشتر برای توسعه سیستم‌های هوشمند و سیستم‌های خبره استفاده می‌شود.

۱-۴-نصب ویژوال پرولوگ

دو نسخه مختلف ویژوال پرولوگ وجود دارد:

۱- نسخه شخصی که مجانی است: Personal Edition

۲-نسخه تجاری: Commercial Version

ابتدا نسخه Personal Edition را از سایت ویژوال پرولوگ به آدرس زیر دانلود کنید:
www.visual-prolog.com/

۱-فایل Setup را اجرا کنید. پنجره ۱-۱ روی صفحه ظاهر می‌شود. روی Next کلیک کنید.

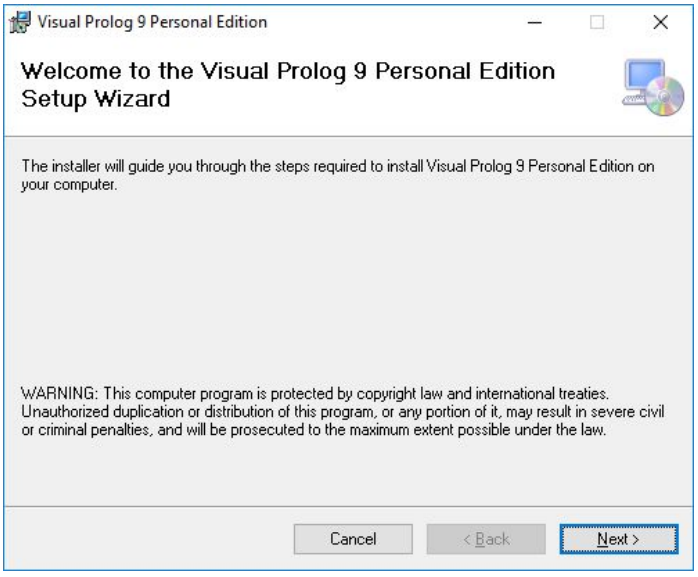
۲-در پنجره ۱-۲ I Agree را انتخاب کنید.

۳-در مرحله بعد(شکل ۱-۳) باید محل نصب برنامه را مشخص کنید . همچنین باید مشخص کنید که فقط شما مجوز استفاده از برنامه را دارید یا اینکه تمام کاربران کامپیوتر شما می‌توانند از برنامه استفاده کنند.

۴-در شکل ۱-۴ آیتم‌هایی که می‌خواهید نصب شود را مشخص می‌کنید.

۵-حال همه چیز برای نصب آماده است. کافی است که در پنجره ۱-۵ روی Next کلیک کنید تا نصب برنامه آغاز شود.

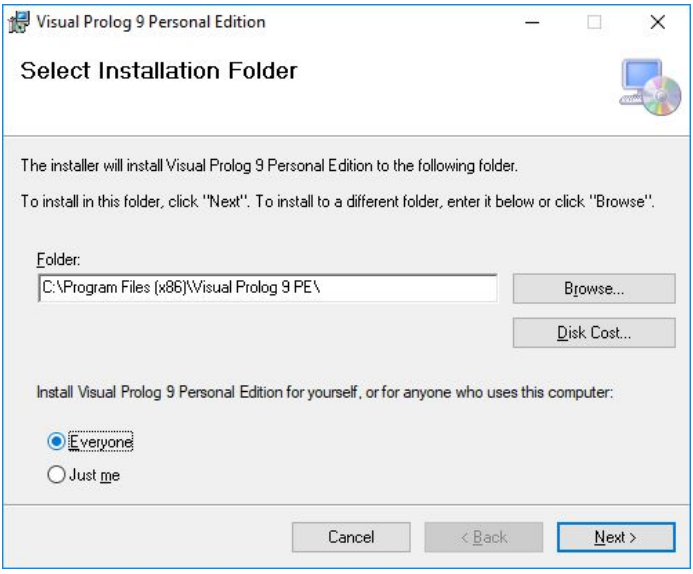
۵-بعد از اتمام نصب پنجره ۱-۶ روی صفحه ظاهر می‌شود که نشان دهنده موفقیت آمیز بودن نصب است.



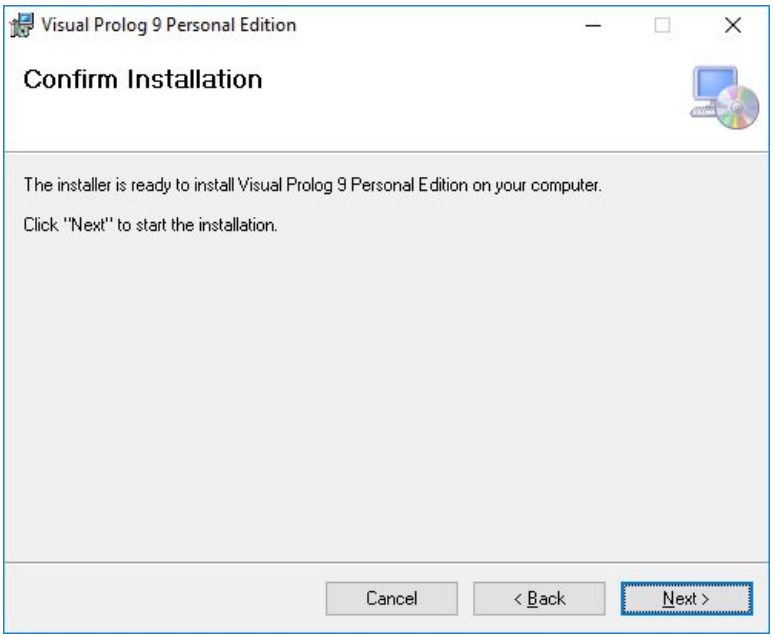
شکل ۱-۱:


Visual Prolog 9 Personal Edition
License Agreement
Please take a moment to read the license agreement now. If you accept the terms below, click "I Agree", then "Next". Otherwise click "Cancel".
LICENSE AGREEMENT
PDC A/S grants and by clicking the "Accept License Agreement" button you accept a license to use the Software and Documentation on these terms. You must return the completed registration card to be eligible for support and updates.
I Do Not Agree
I Agree
Cancel
< Back
Next >

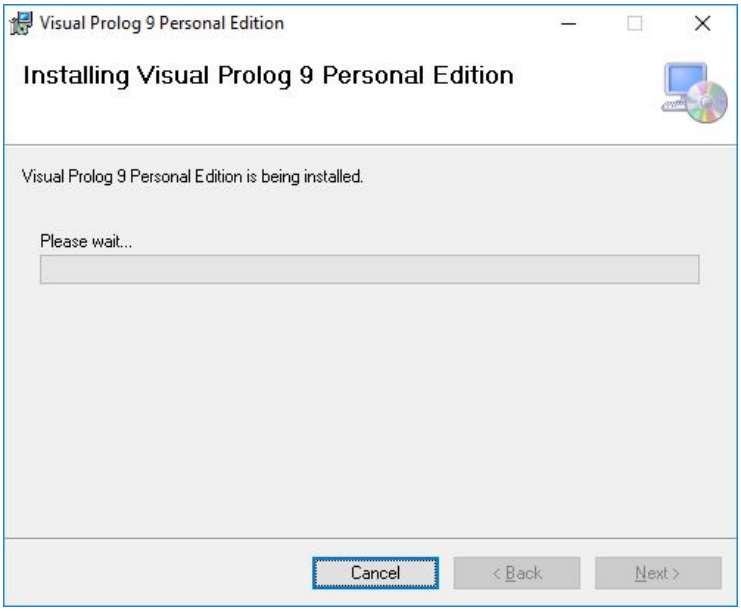
شکل ۱-۲:


Visual Prolog 9 Personal Edition
Select Installation Folder
The installer will install Visual Prolog 9 Personal Edition to the following folder.
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".
Folder:
C:\Program Files (x86)\Visual Prolog 9 PE\
Browse...
Disk Cost...
Install Visual Prolog 9 Personal Edition for yourself, or for anyone who uses this computer:
Everyone
Just me
Cancel
< Back
Next >

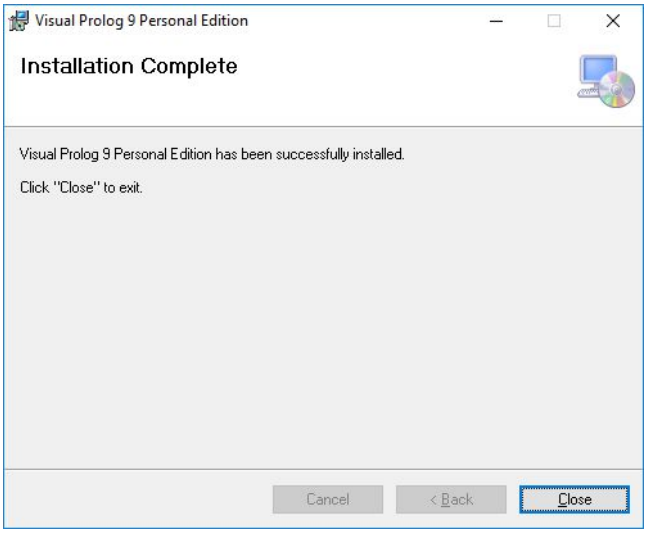
شکل ۱-۳:


Visual Prolog 9 Personal Edition
Confirm Installation
The installer is ready to install Visual Prolog 9 Personal Edition on your computer.
Click "Next" to start the installation.
Cancel
< Back
Next >

شکل ۱-۴:

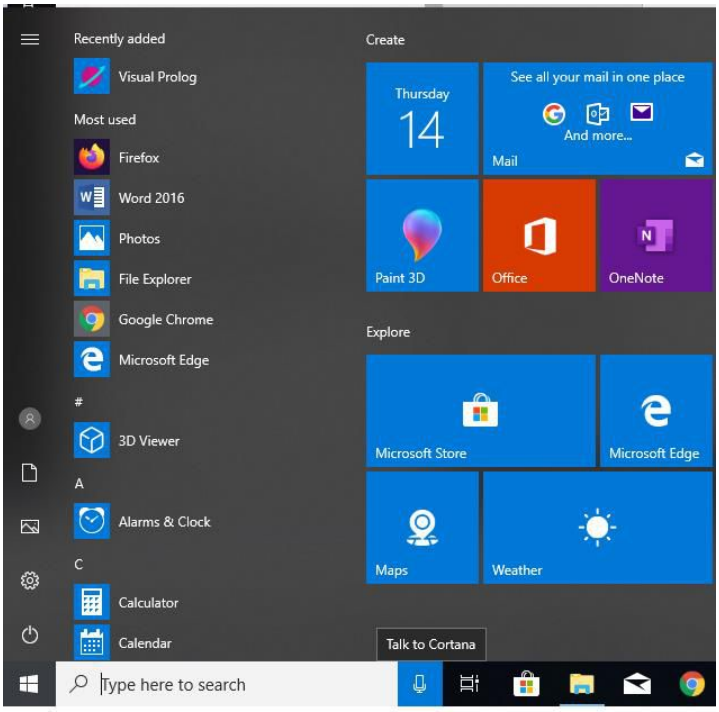

Visual Prolog 9 Personal Edition
Installing Visual Prolog 9 Personal Edition
Visual Prolog 9 Personal Edition is being installed.
Please wait...
Cancel
< Back
Next >

شکل ۱-۵:



شکل ۱-۶:

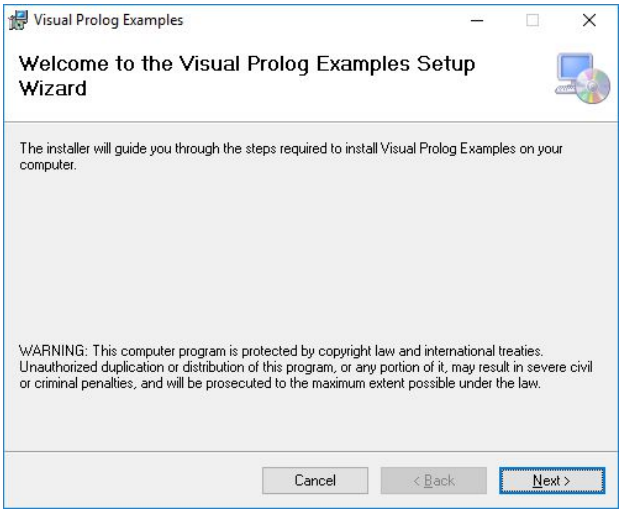
۶- حال از طریق منوی Start ویندوز برنامه را اجرا کنید(شکل ۱-۷).



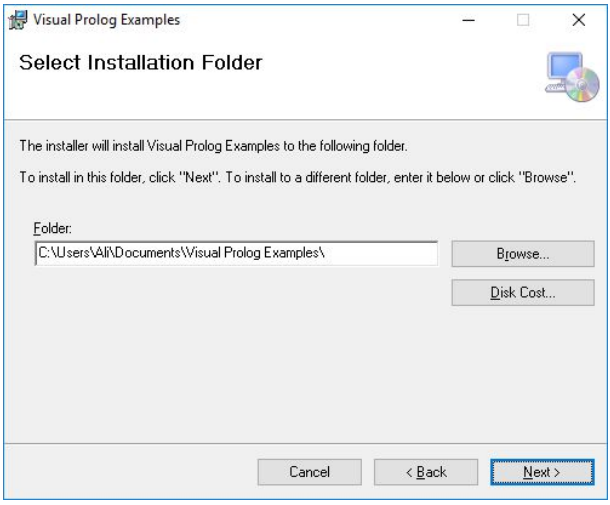
شکل ۱-۷:

۷-اولین بار که برنامه را اجرا می‌کنیم پیغام نصب مثال‌های ویژوال پرولوگ ظاهر می‌شود (شکل ۸-۱). اینها نمونه پروژه‌هایی بسیار مفیدی هستند که به شما در یادگیری ویژوال پرولوگ کمک می‌کنند. روی Next کلیک کتید.

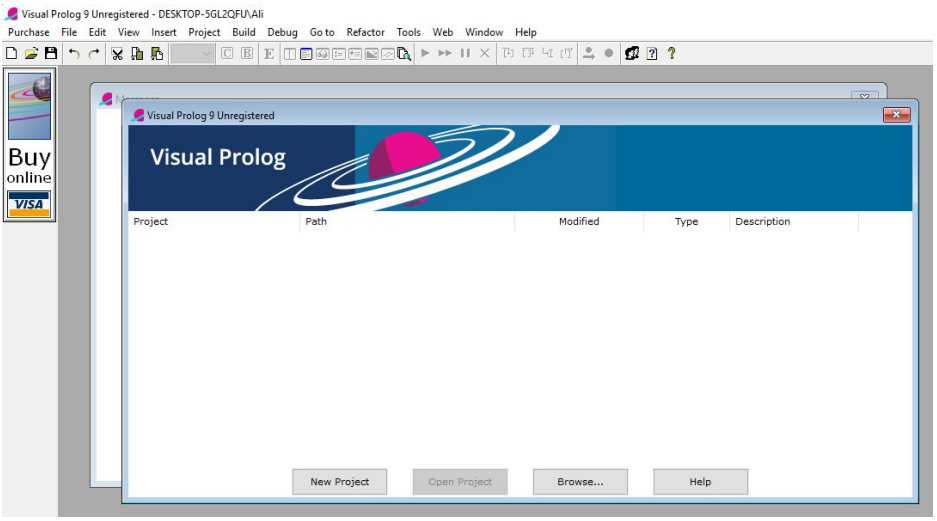
۸- درشکل ۱-۹ باید مسیر نصب مثال‌های ویژوال پرولوگ را مشخص کنید.



شکل ۱-۸:



شکل ۱-۹:



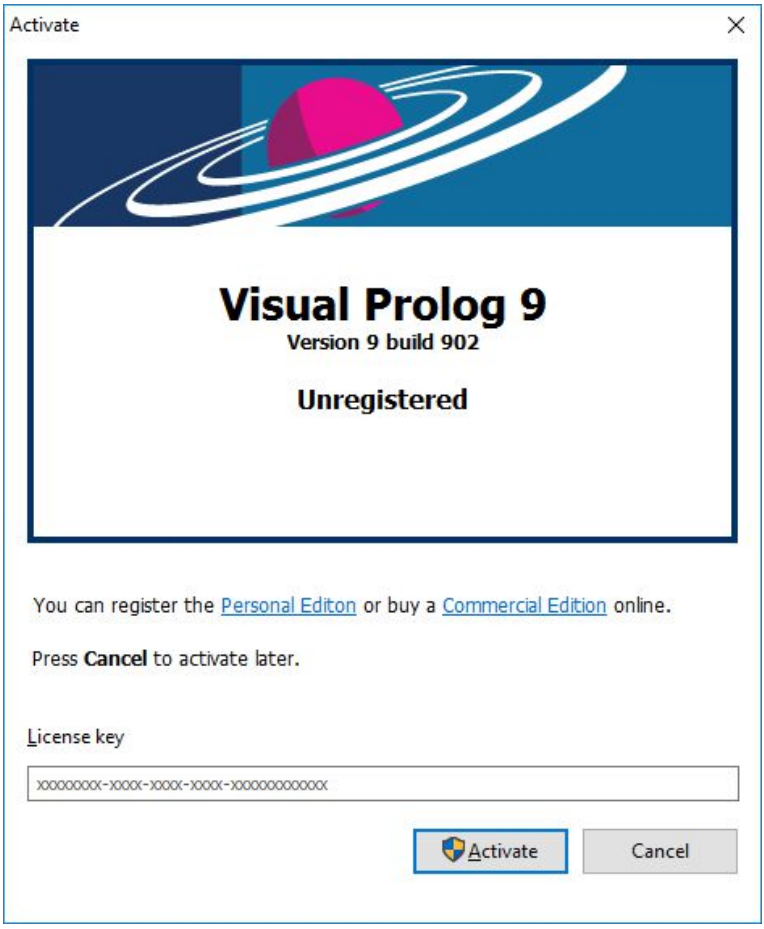
محیط ویژوال پرولوگ

۵-۱-فعال کردن ویژوال پرولوگ

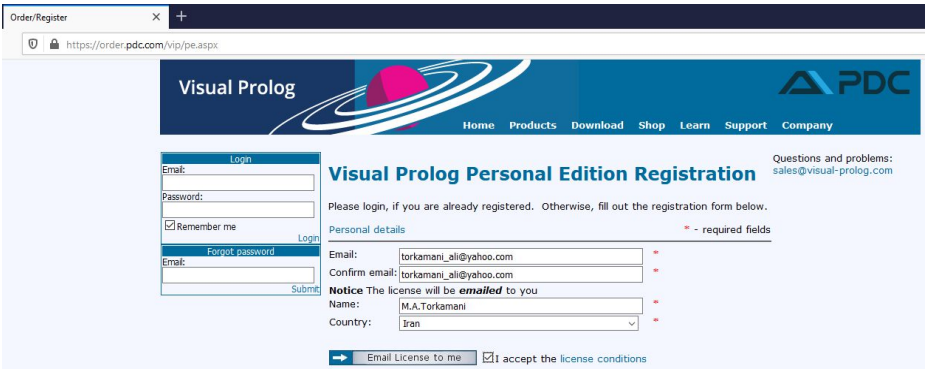
۱-اولین بار که می‌خواهید یک برنامه را کامپایل کنید شکل ۱-۱۰ روی صفحه ظاهر می‌شود. در این پنجره باید روی لینک Personal Edition کلیک کنید. با انجام این کار به وب سایت ویژوال پرولوگ هدایت خواهید شد.

۲- در این مرحله (شکل ۱-۱۱) باید مشخصات و ایمیل خود را وارد کرده و گزینه I Accept the license conditions را تیک بزنید و سپس روی Email license to me کلیک کنید تا کد فعال سازی برای شما ارسال شود.

۳-کد فعال سازی را در پنجره ۱-۱۰ وارد کرده و روی دکمه Activete کلیک کنید.



شکل ۱-۱۰:



شکل ۱-۱۱:

فصل دوم

ساختار برنامه‌های ویژوال پرولوگ

۲-۱-مقدمه

در این بخش ضمن بررسی ساختار برنامه‌های ویژوال پرولوگ، تفاوت آنرا با پرولوگ سنتی بررسی می‌کنیم. به طور کلی یک برنامه ویژوال پرولوگ در موارد ذیل با برنامه پرولوگ تفاوت دارد:

۱-ساختار برنامه متفاوت

۲- امکانات بیشتر و در نتیجه استفاده از فایل‌های متنوع در برنامه

۳- امکانات کنترل Scope

۴- شیء گرایی

۲-۲-اعلان ها و تعاریف

در Prolog، زمانی که ما نیاز به استفاده از محمول (predicate) داریم، به سادگی این کار را بدون هرگونه اعلان انجام می‌دهیم. برای مثال در پرولوگ سنتی grandFather predicate's مستقیماً در قسمت clause نوشته می‌شود تا سر محمول و بدنه آن ایجاد شود. در واقع نیازی نیست که به صراحت در کد برنامه بنویسیم که ساخت یک محمول قرار است بعداً آورده شود.

به طور مشابه، domain در پرولوگ سنتی را می‌توان بدون اعلان و در لحظه ای که به آن نیاز داشته باشید استفاده کنید. اما در ویژوال پرولوگ، قبل از نوشتن کد برای بدنه clause ، برای اولین بار نیاز به اعلام آن به کامپایلر وجود دارد. به طور مشابه، قبل از استفاده از هر دامنه، نیاز است اعلام شود و تا کامپایلر مطلع شود.

از سوی دیگر در بیشتر زبانهای برنامه نویسی اگر *runtime exceptions* یا "استثناء زمان اجرا"، رخ دهد(به عنوان مثال اگر شما به جای یک آرگومان از نوع اعداد صحیح یک عدد حقیقی را وارد کنید) یک خطای زمان اجرا رخ می‌دهد و برنامه متوقف می‌شود. اما در ویژوال پرولوگ اینطور نیست. وقتی که شما محمول ها و دامنه (predicates and domains) را تعریف کردید، چنین نوع گرامر (با آرگومانهای متعلق به دامنه) در دسترس کامپایلر است. بنابراین، هنگامی که ویژوال پرولوگ یک فعالیت تلفیقی(compilation) را انجام میدهد، برنامه را کاملاز لحاظ غلطهای گرامری و سایر موارد چک می‌کند.

به دلیل این ویژگی در ویژوال پرولوگ، بازده کلی یک برنامه نویس را بهبود می بخشد. نیازی نیست که برنامه نویس صبر کنید تا برنامه اجرا شود تا اشکال آن مشخص شود. در واقع، کسانی از شما، که در برنامه نویسی با تجربه هستند، درک آنچه که این مسئله چقدر می‌تواند کمک کند. اغلب، ترتیب و توالی خاص از وقایع است که باعث استثنای زمان اجرا میشوند ممکن است به نوبه خود تا پس از چند سال آشکار نشوند. همه این مسائل نشان می دهد که کامپایلر به دستورالعمل صریح و روشن در مورد محمول‌ها و دامنه(predicates and domains) نیاز دارد که با استفاده از اعلان های مناسب قبل از همان تعریف شده وجود داشته باشد.

۳-۲- کلمات کلیدی ویژوال پرولوگ

در ویژوال پرولوگ کدها در بین چند بخش(sections) با کلمات کلیدی مناسب قرار می‌گیرد تا به کامپایلر اطلاع دهد که کد تولید شده است. مانند کلمات کلیدی زیر:

```
Declarations
predicates
domains.
```

ظاهر شدن یک کلمه کلیدی دیگر به یک بخش جدید در برنامه به معنی اتمام بخش قبلی است. استثنائی برای این قانون وجود دارد و آن کلمات کلیدی "implement" و "end implement" است. کدی که بین این دو کلمه کلیدی است، نشان دهنده کد استفاده شده برای کلاس خاص است. برای درک مفهوم کلاس تصور کنید که آن شبیه به یک ماژول یا بخش از برنامه است.

در ادامه کلمات کلیدی مهم ویژوال پرولوگ را بررسی می‌کنیم.

۱-۳-۲- implement و end implement

در ویژوال پرولوگ، همه کلمات کلیدی دیگر بین این دو کلمه قرار می گیرند. در ویژوال پرولوگ کدی که بین این دو کلمه کلیدی است، مربوط به یک کلاس است. نام کلاس باید بعد از کلمه implement نوشته شود.

open-۲-۳-۲

این کلمه کلیدی برای توسعه scope visibility در کلاس است. برای انجام این کار این کلمه دقیقاً باید بعد از کلمه کلیدی "implement" نوشته شود.

۳-۳-۲-ثابت‌ها

برای تعریف ثابت ها از کلمه کلیدی constants استفاده می‌شود. مثال:

```
constants
   VP = "VisuaL Prolog".
```

بعد از نوشتن این تعریف شما می‌توانید در سرتاسر برنامه از کلمه VP به جای رشته "VisuaL Prolog" استفاده کنید. توجه داشته باشید که انتهای تعریف یک نقطه نوشته می‌شود.

برخلاف متغیرها در پرولوگ، نام ثابتها باید با حرف کوچک شروع شود. ثابت‌های پیش فرض ویژوال پرولوگ در جدول ۲-۱ نمایش داده شده اند. نکته مهم آن است که از ثابت‌ها نمی‌توان به عنوان نام ثابت یا متغیر استفاده شود.

جدول ۲-۱: constant ثابت‌های پیش فرض ویژوال پرولوگ

compilation_date
compilation_time
compiler_buildDate
compiler_version
maxFloatDigits
null
nullHandle
invalidHandle
platform_bits
platform_name

۴-۳-۲- انواع داده اولیه

ویژوال پرولوگ تعداد زیادی انواع داده اولیه دارد که برخی از آنها عبارتند از:

- اعداد صحیح یا Integer- : اعداد صحیح مابین 2147483648- و 2147483647.
- به عنوان نمونه ای از اعداد صحیح می‌توان 1985, 0x0000FAIB, 70, 64, 45, 3 را نام برد. 0x0000FAIB یک عدد صحیح در مبنای شانزده می‌باشد.
- اعشاری یا real : اعداد اعشاری می‌بایست همراه با نقطه اعشار نوشته شوند: .3 ,… 3.4 ,11416 ، 3.1416,2.17E-18 ,6E-19. 1 و غیره.
- رشته یا String: یک رشته از کاراکترها درون علامت (" ") است. مانند , "70.9" "pen", shiraz university, “Ali”و غیره.
- Char: نویسه‌هایی که بین دو علامت کوتیشن تکی (،) قرار می‌گیرند: ‘m’, ‘R’, ‘.’ , ‘ ’ , ‘22’, ‘I’, ‘F’و غیره .
- Symbol یا نماد: مانند . "Na", "Natrium", "K", "Kalium". برای تعریف این نوع از کلمه کلیدی Symbol استفاده می‌شود. ملاحظه می‌کنید که نمادها بسیار شبیه به رشته‌ها می‌باشند: هر دو دنباله‌ای از کاراکترهای یونیکد هستند. با این وجود، به شکلی متفاوتی نسبت به یکدیگر ذخیره می‌شوند. نمادها درون یک جدول جستجو [1] نگهداری می‌شوند و ویژوال پرولوگ به منظور نشان دادن آن نمادها از آدرس آن‌ها در جدول استفاده می‌کند. بنابراین، اگر یک نماد تعداد دفعات زیادی در برنامه‌تان به کار رود، حافظه کمتری را در مقایسه با رشته‌ها مصرف خواهد کرد.

Domains-۲-۳-۵

این کلمه کلیدی دامنه‌هایی که در برنامه استفاده می‌شوند را مشخص می‌کند. نحو یا املای (syntax) مختلفی برای تعریف دامنه وجود دارد که برای انواع مختلف دامنه در کد استفاده می‌شوند.

جدول ۲-۲ Domain های پیش فرض ویژوال پرولوگ را نشان می‌دهد.

1 look-up table

جدول ۲-۲: Domain های پیش فرض ویژوال پرولوگ

any	نوع داده جهانی(Universal)
char	کاراکتر دو بایتی
string	دنباله ای از کاراکترهای مختوم به صفر
string8	دنباله ای از کاراکترهای مختوم به صفرASCII (یک بایتی)
symbol	دنباله ای از کاراکترهای مختوم به صفر
binary	دنباله ای از بایتها
binaryNonAtomic	دنباله ای از بایتها
integer	عدد صحیح علامت دار ۳۲ بیتی
integer64	عدد صحیح علامت دار ۶۴ بیتی
integerNative	عدد صحیح علامت دار با اندازه وابسته به سکو(۳۲ بیت در سکوهای ۳۲ بیتی و ۶۴ در سکوهای ۶۴ بیتی)
unsigned	عدد صحیح بدون علامت ۳۲ بیتی
unsigned64	عدد صحیح بدون علامت ۶۴ بیتی
unsignedNative	عدد صحیح بدون علامت با اندازه وابسته به سکو(۳۲ بیت در سکوهای ۳۲ بیتی و ۶۴ در سکوهای ۶۴ بیتی)
real	عدد ممیز شناور محدوده .-1.7e+308 to 1.7e+308
real32	عدد ممیز شناور
pointer	اشاره گر به یک آدرس از حافظه
handle	یک دستگیره(فایل یا پنجره)
boolean	مقادیر منطقی
factDB	Descriptors به پایگاه داده داخلی نام گذاری شده
compareResult	مقادیر نتایج مقایسه

class facts-۲-۳-۶

این کلمه کلیدی برای تعریف fact هایی که در برنامه استفاده می‌شوند به کار می‌رود. هر fact به وسیله یک نام و آرگومان‌های آن مشخص می‌شود.

class predicates -۲-۳-۷

این بخش شامل تعاریف predicate هایی است که بعداً در بخش clauses تعریف می‌شوند.
predicate ها نیز با استفاده از یک نام و آرگومانهایشان تعریف می‌شوند.
پارامترهای Predicate باید به یکی از روش های زیر تعریف شوند:
۱-استفاده از تعاریفی که در بخش domain استاندارد یا پیش فرض پرولوگ وجود دارد.
۲- تعریف domain خودتان به وسیله بخش domain در برنامه.
domain استاندارد یا پیش فرض شامل ورودی های string، symbol، integer، real یا chatacter و ... است.
مثال:

```
implement main
    open core
domains
day = integer.
month = integer.
year = integer.
auto = car(string, year)
%Compound Domain - car() is a Functor
dealer = string* %List of string
class predicates
date(M month, D day, Y year).
usedCarLot(dealer).
clauses
date(8, 7, 1952).
UsedCarLot([(car(,,Ford,,/ 1982), car("Volvo", 1997), car("BMW", 2005)]).
```

روش دیگر برای تعریف یک Predicate روش as flows برای پارامترهایش است. کلمه anyflow می‌تواند در تعریف predicate های محلی به کار رود.

در ویژوال پرولوگ می‌توانید تعداد پارامترهای ورودی و خروجی را مشخص کنید. در این روش o برای output و i برای input به کار می‌رود:

```
class predicates
date(M month, D day, Y year) (i, i, i) (o, o, o)
```

در ویژوال پرولوگ می‌توانید مشخص کنید که یک جستجو فقط یک راه حل منفرد (single solution) داشته باشد(یعنی deterministic) و یا چندین راه حل ممکن داشته باشد(یعنی non-deterministic). وقتی که یک rule را می‌نویسید، در بخش predicates می‌توانید از کلمات determ یا nondeterm استفاده کنید.

مثال:

```
class Predicates
date(M month, D day, Y year) determ (i, i, i) (o, o, o)
```

predicate های پیش فرض ویژوال پرولوگ در جدول ۲-۳ نمایش داده شده‌اند.

جدول ۲-۳: predicate های پیش فرض ویژوال پرولوگ

and/2 ,/2	predicate_name/1->
assert/1	programPoint/0->
asserta/1	retract/1 nondeterm
assertz/1	retractall/1
bound/1 determ	retractFactDb/1
class_name/0->	sizeBitsOf/1->
compare/2->	sizeOf/1->
convert/2->	sizeOfDomain/1->
digitsOf/1->	sourcefile_lineno/0->
errorExit/1 erroneous	sourcefile_name/0->

fail/0 failure	sourcefile_timestamp/0->
free/1 determ	succeed/0
fromEllipsis/1->	toAny/1->
hasDomain/2 hasDomain/2->	toBinary/1->
in/2 determ in/2 nondeterm	toBoolean/1->
isErroneous/1 determ	toEllipsis/1->
lowerBound/1->	toString/1->
maxDigits/1->	toTerm/1-> toTerm/2->
not/1 determ	tryToTerm/1-> determ tryToTerm/2-> determ
or/2 ;/2	tryConvert/2-> determ
orelse	uncheckedConvert/2->
predicate_fullname/1->	upperBound/1->

نوع بازگشتی در پایان و بعد از علامت <- نوشته می‌شود.

clauses -۲-۳-۸

این بخش شامل تعاریف واقعی predicate هایی است که قبلاً در بخش predicates تعریف شده اند. درواقع یک predicate به وسیله مجموعه‌ای از کلازها تعریف می‌شود. هر کلاز تا زمانی که یکی از آنها succeed شود و یا هیچ کلازی برای اجرا باقی نمانده باشد، ادامه می یابد. اگر هیچ کلازی succeed نشود، fail predicate خواهد شد. اگر یک کلاز succeed شود و کلازهای دیگری در سمت چپ predicate وجود داشته باشد، کنترل برنامه می‌تواند بعداً کلازهای این predicate را به منظور یافتن راه‌حل‌های دیگر backtrack نماید. بنابراین یک predicate می‌تواند fail شود، succeed شود و یا حتی چند بار succeed شود.

goal -۲-۳-۹

این بخش نقطه ورودی اصلی یک برنامه ویژوال پرولوگ را تعریف می‌کند.

۱۰-۳-۲-دسترسی به Scope

ویژوال پرولوگ برنامه را به بخش‌های مختلف تقسیم می‌کند. هر بخش یک کلاس را تعریف می‌کند. هر کلاس در یک فایل جداگانه ذخیره می‌شود. هنگام اجرای برنامه ویژوال پرولوگ به فایل‌های مختلف مراجعه می‌کند. به عنوان مثال فرض کنید pred1 در class1 و pred2 و pred3 در class2 تعریف شده اند. pred1 در داخل clause مربوط به pred2 به صورت زیر فراخوانی می‌شود:

```
clauses
   pred3:-
      ...
      !.

   pred2:-
      class1::pred1,

      pred3,
      ...
```

همانطور که در این مثال دیدید، از نام کلاس به همراه :: برای دسترسی به predicate موجود در آن کلاس استفاده کردیم. به علامت :: ، class qualifier می‌گویند. اما scope visibility pred3 و pred2 یکسان است چون هر دو در یک کلاس تعریف شده اند. به عبارت دیگر در یک Scope تعریف این دو Predicate را جستجو می‌کند.

Scope این دو در یک فایل و بین کلمات کلیدی implement و end implement است.

Scope یک کلاس می‌تواند به وسیله استفاده از کلمه کلیدی open توسعه یابد. این کلمه کلیدی به کامپایلر اطلاع می‌دهد که که نام‌های predicates یا constants یا domains را که در فایل دیگر تعریف شده اند، بیاورد. اگر این کار را انجام دهید، دیگر نیازی به استفاده از :: برای دسترسی به predicates یا constants یا domains موجود در کلاس دیگر نخواهید داشت.

```
open class1
        ...
```

```
clauses
   pred3:-
      ...
      !.

   pred2:-
      pred1, %Note: "class1::" qualifier is not needed anymore, as the scope
area is extended using the 'open' keyword
      pred3,
      ...
```

Cut-۲-۴

فرض کنید شما Rule هایی در برنامه دارید که به صورت غیر قطعی یا non-deterministic هستند، یعنی چندین مسیر مختلف را برای یک راه حل نشان می دهند. در این صورت وقتی که یک راه حل به جواب می رسد باید جلوی جستجوی راه حل های بعدی را بگیرید تا به این ترتیب زمان اجرای برنامه کاهش یابد. در واقع باید جلوی بازگشت به عقب یا backtrack را بگیریم. بنابراین از یک Cut استفاده می کنیم. Cut با علامت تعجب (!) نمایش داده می شود.

! همه backtrack point ها را حذف می کند که شامل تمام نقاط بازگشت به کلاز بعدی و همچنین تمام نقاط بازگشت در کلاز جاری قبل از علامت ! است. .

مثال:

```
clauses
  ppp(X) :-
     X > 7,
     !,
     write("Greater than seven").
  ppp(_X) :-
     write("Not greater than seven").
```

ابتدا یک backtrack point روی کلاز دوم ایجاد می‌کند و سپس اولین کلاز اجرا می‌شود. اگر نتیجه تست X > 7 موفقیت آمیز باشد، به ! می‌رسد که باعث می‌شود backtrack point روی کلاز دوم حذف شود.

مثال:

```
clauses
  ppp() :-
     qqq(X),
     X > 7,
```