معاونت پژوهشی

گزارش نهایی طرح پژوهشی

عنوان طرح :

طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی

ParsPL

مجری : احمد یوسفان

همکار(همکاران): .....

.....

تاریخ آغاز : ۱۳۸۹/۱۲/۲۴ **تاریخ پایان : ۱۳۹۰/۶/۸**

معاونت پژوهشی

گزارش نهایی طرح پژوهشی با عنوان

طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی

مجری : احمد یوسفان

همکار(همکاران):.....
.....

این طرح پژوهشی با حمایت و پشتیبانی معاونت پژوهشی دانشگاه کاشان اجرا شده است

پیشنهاد طرح فوق در تاریخ ۱۳۸۹/۶/۱ به تصویب شورای پژوهشی دانشکده مهندسی رسید و در تاریخ گزارش نهایی آن مورد تصویب قرار گرفت.

مجید منعم زاده

معاون پژوهشی

سپاسگزاری

از معاونت پژوهشی دانشگاه کاشان، جناب آقای دکتر مجید منعم‌زاده و معاون پژوهشی دانشکده جناب آقای دکتر عبدالله زاده و کارمندان حوزه معاونت محترم پژوهشی دانشگاه کاشان به خاطر پشتیبانی از این طرح کمال سپاسگزاری را دارم

چکیده

زبان‌های برنامه نویسی گوناگونی تا کنون ساخته شده و به کار گرفته شده است. فهرست بزرگی از این زبان‌ها می‌توان ارائه نمود که بسیاری از آن‌ها امروزه دیگر کاربردی ندارند یا بسیار کم به کار برده می‌شوند. هر کدام از این زبان‌ها به دلیل‌های گوناگونی به وجود آمده‌اند و کاربردها و توانایی‌های ویژه‌ای را داشته‌اند. تا کنون کوشش‌هایی برای فارسی سازی زبان‌های برنامه نویسی متداول انجام شده است ولی تا جایی که نگارنده می‌داند زبان برنامه نویسی‌ای که از پایه ساخته شده باشد و فقط فارسی شده‌ی زبان دیگری نباشد و ویژگی‌های زبان فارسی را داشته باشد تا کنون ارائه نشده است. بنابراین در این طرح پژوهشی کوشش شد که زبان برنامه نویسی همه منظوره‌ای پیشنهاد شود که بتوان به زبان‌های گوناگون دنیا و از جمله زبان فارسی با آن برنامه نویسی کرد. همچنین ویژگی‌های زبان فارسی در آن به خوبی در نظر گرفته شده است به گونه‌ای که یک فارسی زبان به سادگی بتواند به این زبان برنامه نویسی برنامه بنویسد. این زبان دربردارنده‌ی ایده‌های جدیدی است که یک زبان برنامه نویسی همه منظوره و همزمان بسیار ساده باید دربرداشته باشد. پیاده‌سازی مفسر زبان REXX در سال ۱۳۷۸، پیشنهاد شبه زبان برنامه نویسی برای دستور زبان فارسی در سال ۱۳۸۲، پیشنهاد و همکاری در پیاده‌سازی مفسر زبان جبر رابطه‌ای تحت وب در سال ۱۳۸۷ از جمله کارهایی است که نگارنده‌ی این پروژه پیش از این در زمینه زبان‌های برنامه نویسی و نوشتن کامپایلر و مفسر انجام داده است.

فهرست مطالب

فهرست

فهرست

فصل یکم

تاریخچه‌ای از زبان‌های برنامه نویسی

۱. تاریخچه‌ای از زبان‌های برنامه نویسی

انسان‌ها همواره می‌کوشیده‌اند ابزارهایی برای ساده سازی کارهای خود بسازند تا به کمک آن به گمان خود بتوانند زندگی آسوده‌تری را فراهم کنند. رایانه یکی از ابزارهای ساخت انسان است که برای ساده‌تر شدن انجام کارها آماده شده است. امروزه رایانه‌ها کاربردهای بسیار گوناگونی دارند و در بسیاری از جاها به کار برده می‌شوند. همچنین گونه‌های بسیاری از رایانه‌ها امروزه ساخته شده است و به کار گرفته می‌شود. برخی از آن‌ها بسیار ویژه هستند و فقط برای کاربردهای ویژه‌ای به کار برده می‌شوند و برخی همه منظوره بوده و برای بسیاری از کاربردها می‌توان آن‌ها را به کار برد. به کارگیری رایانه در اجتماع انقلاب بزرگی را به وجود آورده است که برخی بر این باورند که ارزش این دگرگونی بیشتر از انقلاب صنعتی است. برای ارتباط ساده‌تر با رایانه‌ها زبان‌های برنامه نویسی به وجود آمدند.

در این فصل کوشیده می‌شود که تاریخچه‌ی گذرایی از زبان‌های برنامه نویسی بدون پرداختن به جزییات زبانی آن‌ها آورده شود.

۱.۱. الگوریتم

برای حل مسآله به کمک رایانه باید در آغاز الگوریتم کاری که باید انجام شود به گونه‌ای در اندیشه یا به صورت نوشته بر روی کاغذ مشخص شده باشد درواقع در زندگی روزمره ما نیز بارها و بارها الگوریتم‌ها را به کار می‌بریم یا آن‌ها را به دیگران می‌گوییم. کلمه‌ی الگوریتم برگرفته شده از نام ریاضیدان بزرگ ایرانی در قرن دوم هجری شمسی، ابوجعفر بن محمد بن موسی الخوارزمی، است. هر دستورالعملی که مرحله‌های گوناگون انجام کاری را به زبان دقیق و با جزییات کافی بیان نماید، به طوری که ترتیب مرحله‌ها و شرط پایان کار در آن به خوبی مشخص شده باشد الگوریتم نامیده می‌شود.

۲.۱. روند نما (flowchart)

روند نما روشی گرافیکی برای نمایش یک الگوریتم است. روند نما این کمک را می‌کند تا یک نمایش بصری از الگوریتم فراهم شود. همچنین بیش از چهل سال پیش استانداردی برای کشیدن روندنماها آماده شده است.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

روندنماها نمی‌توانند مستقیم روی کامپیوتر اجرا شوند و فقط ابزاری نمایشی و اغلب آموزشی برای آموزش برنـامه نویسی هستند.

۳.۱. نخستین طرح برنامه نویسی

چرتکه شاید نخستین ماشین محاسباتی در جهان باشد که قرن‌ها برای انجام محاسـبه‌های مـالی بـه کـار گرفتـه می‌شد. دانشمندان گوناگونی برای ساخت دستگاهی که بتوانـد محاسـبات را انجـام دهـد کوشـیدند از آن جملـه می‌توان به جورج و ادوارد شوتز، بلز پاسکال، چارلز بابیج و جوزف کلمنت اشاره کرد. ماشین‌های بافنده‌ی ژاکـارد نخستین ماشین‌هایی بودند که به کمک الگویی که بر روی یک کارت پانچ گذاشته شده بود عمل بـافتن را انجـام می‌داد. از این دیدگاه این ماشین‌های بافنده را می‌توان نخستین ماشین‌هایی دانست که برنـامه‌ی ذخیـره شـده را اجرا می‌کردند. بر این پایه بابیج پیشنهاد ساخت یک موتور محاسباتی را داد. Ada که در یـک مهمـانی بـا بابیـج آشنا شده بود پیش‌بینی کرد که این ماشین می‌تواند به ساخت موسیقی، ساخت گرافیک و حل مسأله‌های علمـی و ریاضی بپردازد. همچنین او طرحی را پیشنهاد داد که به کمک این ماشین بتـوان عـددهای برنـولی را محاسـبه کرد. بنابراین کار او را به نوعی می‌توان نخستین کوشش برای نوشتن یک زبان برنامه نویسی دانست. به افتخـار او دو قرن بعد نام او بر یک زبان برنامه نویسی نهاده شد.

۴.۱. Plankalkul نخستین زبان برنامه نویسی سطح بالا

کونارد زوسه(Konrad Zuse) در ۱۹۴۶ میلادی نخستین زبان برنامه نویسی سطح بالای دنیا را پیشـنهاد داد. زوسه مهندسی آلمانی بود که دستگاه Z1 را طراحی کرد و به صورت عملی آن را پیاده‌سازی کرد که می‌توان آن را نخستین رایانه‌ی دنیا نامید. درواقع دستگاه Z4 از او را می‌توان یک رایانه‌ی کامل (نسبت بـه رایانه‌هـای اولیـه پس از آن) دانست. زبان Plankalkul یک زبان بسیار پیشـرفته نسـبت بـه زمـان خـود بـود کـه دربردارنـده‌ی توانایی‌های بسیار خوب برنامه نویسی بود. او در همان زمان (۱۹۴۶) برنامه‌هایی نیز به این زبـان نوشـت. ایـن در حالی بود که در همان زمان رایانه‌ی ENIAC برای اینکه بتواند مسأله‌ی تازه‌ای را حل کنـد بایـد سیم‌کشـی‌اش عوض می‌شد. این زبان دارای انتساب متغیر، حلقه، شرط، نوع‌هـای پـایه، عمل‌هـای محاسـباتی، تابع‌هـایی بـرای پردازش لیست و مجموعه بود. این زبان توانایی بازگشتی(recursive) را نداشت و جالب اینکه دسـتور GOTO نیز در زبان وجود نداشت. یادآوری می‌شود که دستور goto تا دهه‌ها بعد به کارگرفته شد و به کارگیری آن یکی از بزرگترین اشتباهاتی بود که برنامه نویسان انجام می‌دادند.

در آن زمان کامپایلری برای این زبان ساخته نشد ولی ایده‌های آن در زبان‌های پس از آن به کار برده شد. در سال ۲۰۰۰ میلادی کامپایلری برای زیر مجموعه‌ای از این زبان در دانشگاه آزاد برلین ساخته شـد و نخسـتین برنامه‌های این زبان ۵۵ سال پس از معرفی آن اجرا شدند.

۵.۱. سخت‌افزار برای برنامه ریزی

نخستین رایانه‌های واقعی به گونه‌ای بودند که برای انجام یک کار باید سیم کشی آن‌ها تغییر می‌کرد و این کار برنامه‌ریزی آن‌ها را برای حل یک مسأله بسیار دشوار می‌کرد. بنابراین دستگاه‌هایی ساخته شدند که کارشان گرفتن ورودی از کاربر و تبدیل آن به شکل دلخواه سخت‌افزار بود تا کاربر با سادگی بیشتری بتواند کار خود را انجام دهد. این دستگاه‌ها دربردارنده‌ی ده‌ها یا صدها دکمه بودند تا بتوانند برنامه‌ریزی را انجام دهند. زوسه برای رایانه‌ی ساخت خودش به نام Z4 یک چنین دستگاهی را به نام programator طراحی کرد ولی عملاً آن را نساخت. به دلیل دشواری‌های فراوانی که کار با رایانه‌های قدیمی داشت ایده‌های جالبی را کاربران آن برای ساده‌تر شدن کارشان می‌دادند که عملاً باعث شد که در نسخه‌های بعدی سخت‌افزارها آن ایده‌ها پیاده‌سازی شوند. برای نمونه به کار گیری چند باره‌ی یک کار درون یک نوار (tape) که به نظر کار ساده‌ای است را نخستین بار شخصی انجام داد و آن را در بوق و کرنا کرد. کار با رایانه‌های قدیمی آن‌چنان دشوار بود که به گفته‌ی برخی افزون بر سپاسگزاری از طراحان و سازندگان رایانه‌های اولیه در آن زمان باید از کاربران آن‌ها نیز یاد کرد که به چه سختی‌ای با این دستگاه‌ها کار می‌کردند. حتی تا سال‌ها بعد روی رایانه‌های بزرگ ۷۰۹۴ از شرکت IBM لامپ‌های کوچکی (LED) قرار داشت که وضعیت صفر یا یک بودن بیت‌های ثبات‌های رایانه را نشان می‌داد تا هنگام پیش آمدن هر گونه توقف یا مشکل دیگر متصدی بخت برگشته با بررسی این بیت‌ها مشکل را بیابد.

در سال ۱۹۵۲ Heinz Rutishauser آلمانی در هنگام به کارگیری Z4 به این حقیقت پی‌برد که یک رایانه‌ی همه منظوره خودش می‌تواند برنامه‌ریزی شود بدون اینکه نیاز باشد وسیله‌های دیگری برای برنامه‌ریزی آن به کار برده شوند. پیشنهاد برنامه‌ی ذخیره شده از سوی John Von Neumann راه گشای بسیاری از مشکلات شد. به این ترتیب که برنامه در کنار داده‌های آن در یک مکان(حافظه‌ی داخلی رایانه) جاگرفته و سپس به کمک یک شمارنده‌ی برنامه دستورها یکی پس از دیگری اجرا شوند. شاید امروزه این کار بسیار ساده و پیش پا افتاده به نظر آید و ساده‌ترین حالتی است که اکنون اغلب در آغاز کتاب‌های معماری رایانه به عنوان طرح ساده‌ی یک رایانه نوشته می‌شود.

نخستین رایانه‌ای که ایده‌ی برنامه‌ی ذخیره شده را به کار برد EDSAC نام داشت که در کمبریج انگلستان ساخته شد. D. J. Wheeler یکی از عضوهای EDSAC روشی را اختراع کرد که در آن آدرس برنامه اصلی (main) در جایی ذخیره شود و برای اجرای برنامه به آن بخش آغاز برنامه پرش انجام شود. این روش که پرش Wheeler نامیده شد پیش زمینه‌ی فراخوانی‌های تابع امروزین است.

۶.۱. زبان ماشین

به جز شرکت‌های سازنده‌ی سخت‌افزار و توسعه دهندگان برخی از نرم‌افزارهای حیاتی مانند سیستم عامل (آن هم فقط برای بخش‌های کمی از آن) نوشتن مستقیم صفر و یک در یک رایانه همه کاره همچنین برنامه نویسی با این روش برای یک رایانه‌ی بزرگ چندان انجام نشد. بنابراین برنامه نویسی مستقیم به زبان ماشین بر روی

سخت‌افزار کاری بود که در بدترین حالت به صورت غیر مستقیم و به کمک سوراخ کردن کارت‌های پانچ انجام می‌شد. در واقع پس از سیم کشی در رایانه‌های نخستین برای برنامه‌ریزی آن‌ها دستگاه‌های کارت خوان به کار گرفته شد تا نیازی به نوشتن مستقیم در ثبات‌های رایانه نباشد. در آن زمان‌ها هنوز مفهوم حافظه اصلی به وجود نیامده بود ولی خواندن ثبات‌ها در هنگام پیش آمدن مشکل متداول بود.

این در حالی است که در دنیای ریزکنترل‌گرها (micro controller) حتی تا یک دهه پیش نیز این کار متداول است و هنوز برای برخی از نوع‌های آن این کار انجام می‌شود. به این صورت که به کمک یک keypad ساده‌ی شانزده شانزدهی این ریزکنترل‌گرها برنامه‌ریزی می‌شدند و کدهای شانزده شانزدهی زبان ماشین مستقیم در حافظه‌ی آن‌ها نوشته می‌شد. البته امروزه بسیاری از این ریزکنترل‌گرها به کمک ارتباط با رایانه و با یک زبان برنامه نویسی مانند C برنامه‌ریزی می‌شوند.

۷.۱. زبان اسمبلی

کار کردن با دستورهای زبان ماشین با صفر و یک یا مبنای شانزده بسیار دشوار است ساده‌تر است که به جای کار کردن به زبان ماشین نام‌ها یا نمادهایی جایگزین دستورهای ماشین شود و این نام‌ها و نمادها برای نوشتن برنامه به کار برده شود. بنابراین زبان «اسمبلی» (Assembly) ساخته شد که دستورهای زبان ماشین (صفر و یک) را به دستورهای نمادین نگاشت می‌کند.

برنامه‌ای که به زبان اسمبلی نوشته می‌شود به شکل بسیار ساده‌تری در دو گذر به زبان ماشین نگاشته می‌شود. در گذر یکم دستورهای نمادین با مقدارهای عددی زبان ماشین خود جایگزین می‌شوند و در گام دوم آدرس‌های پرش و دیگر آدرس‌ها مقدار دهی می‌شوند. برنامه‌ای که این نگاشت را انجام می‌دهد، «اسمبلر» (Assembler) نامیده شد. البته گسترش‌هایی نیز در زبان‌های اسمبلی داده شد که بتوان کارهای پیچیده‌تری را نیز انجام داد. برای نمونه در زبان اسمبلی می‌توان «ماکرو» (macro) نوشت که هر ماکرو جایگزین چندین دستور زبان ماشین می‌شود. برخی از اسمبلرها کتابخانه‌ی بزرگی از ماکروها را فراهم می‌کنند تا برنامه نویسی به زبان اسمبلی ساده‌تر گردد.

جالب این است که ساخته شدن و گسترش یافتن اسمبلرها و زبان اسمبلی در بیرون از شرکت‌های رایانه‌ای انجام شد. برای نمونه گروهی از مشتریان شرکت IBM گروهی را به نام SHARE تشکیل دادند که اسمبلر ویژه‌ی خودشان را بر روی رایانه‌های IBM گسترش دادند و در طول زمان کتابخانه‌ی بزرگی از ماکروها را برای آن آماده کردند. بنابراین به کارگیری زبان اسمبلی برای برنامه نویسی پس از به کارگیری زبان‌های بسیار ساده‌ی برنامه نویسی در آن زمان انجام شد. زیرا کاربران نمی‌توانستند منتظر بمانند تا زبان مناسب همراه با کامپایلر مناسب آن برای سخت افزاری که خریداری کرده‌اند آماده شود تا بتوانند نیازهای خود را برآورده کنند همچنین نرم افزارهای روی رایانه‌های خریداری شده نیز کارهای ساده‌ای را انجام می‌دادند.

همین نکته نشان می‌دهد که مشکلات زبان‌های برنامه نویسی و مهم‌تر از آن کامپایلرها در آن زمان تا چه

اندازه بوده است که مشتریان مجبور به انجام چنین کاری شدند. البته برای شرکت IBM کوشش‌های گروه SHARE بسیار سودمند بود زیرا باعث می‌شد امکانات تازه‌تری برای رایانه‌های تازه‌تر IBM آماده شود بدون اینکه هزینه‌ای از سوی شرکت پرداخت شود و مشتریان بیشتری به دلیل ساخته شدن امکانات جدید جذب رایانه‌های IBM شدند. برای نمونه فروش خوب رایانه‌ی IBM ۷۰۴ به همین دلیل بود. همچنین کارهای این گروه باعث شد که راه پیشرفت و گسترش رایانه‌ها و نیازهای آن‌ها نیز مشخص شود.

عضوهای گروه SHARE این امکان را داشتند که رایگان کدهای یکدیگر را به کار ببرند و همچنین آن کدها را تغییر داده و به دلخواه خود به کار ببرند. این گروه برای اینکه خود را به عنوان رقیبی برای شرکت‌های رایانه‌ای نشان ندهند نخستین جلسه‌ی بزرگ خود را در شرکت RAND برگزار کردند. شاید این گروه نخستین گروه از دسته گروه‌های توسعه دهندگان آزاد نرم‌افزار بودند که بیرون از شرکت‌های رایانه‌ای به وجود آمدند و سهم بزرگی در همه گیر شدن رایانه‌ها داشتند. شاید بتوان جنبش‌های متن باز (open source) کنونی را به نوعی ادامه دهندگان این راه دانست. امروزه گروه‌ها و جامعه‌های (community) گوناگونی در دنیای رایانه برای هدف‌های بسیار گوناگونی به وجود آمده است که اغلب آن‌ها برای انجام کاری (اغلب نرم‌افزاری) با هم همکاری می‌کنند.

۸.۱. نخستین مفسر و کامپایلر

Laning و Zierler در سال ۱۹۵۴ میلادی نخستین کامپایلر را برای زبان برنامه نویسی بسیار ساده‌ای ساختند. این زبان بسیار ساده برای انجام محاسبه‌های ساده طراحی شده بود و یک زبان کامل نبود. آن‌ها نامی را بر روی زبان برنامه نویسی خود و کامپایلرش نگذاشتند. کامپایلری که برای این زبان ساخته بودند؛ بسیار کامل بود و توانایی‌های خوبی داشت به گونه‌ای که برخی از مفهوم‌هایی به کارگرفته شده در پیاده‌سازی این کامپایلر در چند نسل پس از آن به کار گرفته شد.

Hopper نوع بسیار ساده‌ای از برنامه‌های را که به کار می‌برد کامپایلر می‌نامید که نمی‌توان امروزه آن‌ها را کامپایلر نامید زیرا کار آن همانندی چندانی با کامپایلرها نداشت. به هر حال او به یکسری از کارهای ساده‌ای نیز که باعث خودکار سازی و ساده‌سازی روند اجرای برنامه‌ها می‌شد کامپایلر می‌گفت. به این ترتیب از ۱۹۵۲ به این طرف نام‌های A-0 تا A2 تا ۱۹۵۳ در رایانه‌ی UNIVAC به چشم می‌خوردکه نام کامپایلر را یدک می‌کشند. بنا به تعریف Hopper کامپایلر برنامه‌ی ساده‌ای بود که یک تکه کد را در مکان مناسبی در حافظه که کاربر می‌خواست، کپی می‌کرد. روشن است که چنین تعریفی به هیچ عنوان با مفهومی که از کامپایلر با آن آشنا هستیم یکی نیست بلکه بسیار نیز از آن فاصله دارد. البته وظیفه‌ی او بیشتر چیزی بود که امروز باید در بهترین حالت نام آن را متصدی (operator) رایانه بگذاریم برای همین نیز چنین کاری را بزرگ می‌پنداشت و خودکار سازی این کارهای ساده را کامپایلر می‌نامید.

اصولاً Hopper علاقه‌ی فراوانی به مصاحبه، به ویژه از نوع تلویزیونی آن و بزرگ نمایاندن چیزها داشت و

همزمان علاقه‌ی وافری به درجه‌های دریاداری در ارتش ایالات متحده‌ی آمریکا داشت به گونه‌ای که زندگی خانوادگی‌اش را برای رسیدن به این هدف («پوچ» مانند بسیاری دیگر از هدف‌های ما آدم‌های پوچ) گذاشت!!! او بالاخره در هنگام بازنشستگی توانست به درجه‌ی دریاسالاری تمام برسد و آن را با خود به «گور» ببرد. به گمان برخی او همچنین توانست سطرهایی از تاریخ رایانه را نیز برای خود کند و خود را در تاریخ رایانه ماندگار کند. پرسش اینجاست که این سطرهایی از تاریخ رایانه یا کل تاریخ گیتی نیز اگر به نام کسی باشد چه سودی برایش پس از مرگ دارد؟ این پرسشی است که هنوز پاسخی برای آن نیافته‌ام و پوچ‌تر آنکه چرا برخی برای رسیدن به چنین چیزی حاضرند هر کاری بکنند. نویسنده‌ی این گزارش تا کنون نشخوارهای فلسفی گوناگونی، از کتاب گرفته تا نوشته‌های پراکنده در زمینه‌ی سودمندی و ارزش چنین چیزی خوانده است که به هیچ عنوان نمی‌خواهد نامی از آن نوشته‌ها یا نویسندگان پوچ‌شان ببرد ولی با این همه نه تنها دلیل‌های پیچیده و سنگین! ایشان پاسخی را دست و پا نکرده است بلکه بدگمانی نسبت به آنان را نیز در نویسنده افزایش داده است.

۹.۱. زبان فرترن (FORTRAN نخستین زبان برنامه نویسی به کارگرفته شده)

این زبان برنامه نویسی دومین زبان برنامه نویسی دنیا است که ساخته شده است و نخستین زبان برنامه نویسی است که برای آن مترجم (کامپایلر) کاملی نوشته شد و این زبان برای کار برنامه نویسی به کار گرفته شد. فرترن (FORmula TRANslation) از سوی IBM در ۱۹۵۷ برای رایانه‌ی ۷۰۴ این شرکت ارائه شد. از همان آغاز مشتریان به سوی این زبان کشیده شدند و به طور گسترده‌ای آن را به کار بردند. نزدیک بودن این زبان به جبر معمولی به جا افتادن آن بسیار کمک کرد.

در آن زمان نرم‌افزار هنوز بسیار جدی گرفته نشده بود و سخت‌افزار ارزش فراوانی داشت ولی با گذشت زمان وضعیت تغییر کرد و نرم‌افزار ارزشمند شد. در سال ۱۹۹۳ آکادمی ملی مهندسی آمریکا جایزه‌ی Charles Stark Draper خود را به John Backus برای ساخت و گسترش زبان فرترن داد. روشن است که چقدر طول کشید که زبان‌های برنامه‌نویسی به عنوان بخش بسیار ارزشمندی از دنیای رایانه در نظر گرفته شوند. در این میان زبان‌های برنامه‌نویسی گوناگونی به وجود آمدند و برخی از آن‌ها باقی ماندند. زبان فرترن با همه‌ی دیرینه‌ای که دارد امروزه هنوز هم زیاد به کار برده می‌شود. توضیح کوتاهی از زبان فرترن در فصل بعد گذاشته شده است.

تاریخچه‌ای از زبان‌های برنامه نویسی در سایت زیر گذاشته شده است.

http://www.levenez.com/lang/

فصل دوم

فشرده‌ای از برخی از زبان‌های بررسی شده

۲. فشرده‌ای از برخی از زبان‌های بررسی شده

زبان‌های برنامه نویسی بسیار زیادی وجود دارد که برخی از آن‌ها امروزه متداول هستند و به طور عمومی بـه کـار برده می‌شوند. در این پژوهش تعداد زیادی از زبان‌های برنـامه نویسـی بررسـی شـدند و در ایـن فصـل برخـی از ویژگی‌ها و دستورهای تعدادی از این زبان‌های برنامه‌نویسی نوشته شده است. برخی از زبان‌هـا ماننـد C و C++ و C# و java و Pascal و Basic و Prolog و PHP و ASP و Matlab و Python یا همانند آن که اغلب برای برنامه نویسان رشته‌ی کامپیوتر آشنا هستند در این فصل آورده نشده است.

۱.۲. زبان فرترن

همان‌گونه که پیش از این گفته شد زبان فرترن زبانی است که به عنوان یک زبان همه منظوره به کار گرفته شد. این زبان نسخه‌های گوناگونی دارد که با گذر زمان به روز شده‌اند. در زیر ساده‌ترین نمونه به این زبان نوشته شده است.

```
program HelloWorld
 write (*,*) 'Hello, world!'    ! This is an inline
comment
end program HelloWorld
```

این زبان آنقدر در کارهای گوناگون به ویژه در کارهای پژوهشی و علمـی بـه کـار گرفتـه شـد کـه هنـوز در بسیاری از رشته‌های علمی به عنوان زبان اول به کار برده می‌شود. کتابخانه‌های بسیار گوناگون و بزرگی برای ایـن زبان نوشته شده است. این زبان دارای نسخه‌های رسمی ۲، ۳ ، ۴، ۶۶، ۷۷ ، ۹۰ ، ۹۵ ،۲۰۰۳، ۲۰۰۸ است.

در زیر کارکرد حلقه در این زبان نشان داده شده است.

```
 outer: DO
  inner:    DO i = j, k, l       ! from j to k in steps of
l (l is optional)
       :
       IF (...) CYCLE
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
      :
      IF (...) EXIT outer
     END DO inner
    END DO outer
```

این زبان توانای‌های جالبی برای کار با آرایه‌ها و ماتریس‌ها دارد. برای نمونه خط زیر جمع بخشی از یک آرایه را نشان می‌دهد.

```
tot = SUM( a(m:n) )
```

در فرترن ۲۰۰۳ می‌توان اشاره‌گری به بخشی از یک آرایه تعریف کرد. در دستور زیر window که اشاره‌گری به بخشی از table است به r:r+n-m , s:s+q-p اشاره می‌کند.

```
window(r:,s:) => table(m:n,p:q)
```

در برنامه‌ی زیر کار یک حلقه‌ی تکرار در این زبان نشان داده شده است.

```
! sum.f90
! Performs summations using in a loop using EXIT
statement
! Saves input information and the summation in a data
file

program summation
implicit none
integer :: sum, a

print*, "This program performs summations. Enter 0 to
stop."
open(unit=10, file="SumData.DAT")
sum = 0
do
    print*, "Add:"
    read*, a
    if (a == 0) then
     exit
    else
     sum = sum + a
    end if
    write(10,*) a
end do
print*, "Summation =", sum
write(10,*) "Summation =", sum
close(10)
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
end
```

خروجی برنامه در پایانه زیر نشان داده شده است.

```
This program performs summations.  Enter 0 to stop.
Add:
1
Add:
2
Add:
3
Add:
0
Summation = 6
```

و پرونده‌ی sumData.dat در بردارنده‌ی خط‌های زیر خواهد بود.

```
1
2
3
Summation = 6
```

برنامه‌ی زیر مساحت یک استوانه را محاسبه می‌کند.

```
program cylinder
! Calculate the surface area of a cylinder.
! Declare variables and constants.
! constants=pi
! variables=radius squared and height
implicit none    ! Require all variables to be
explicitly declared
 integer :: ierr
 character(1) :: yn
 real :: radius, height, area
 real, parameter :: pi = 3.141592653589793
 interactive_loop: do
!   Prompt the user for radius and height
!   and read them.
  write (*,*) 'Enter radius and height.'
  read (*,*,iostat=ierr) radius,height
!   If radius and height could not be read from input,
!   then cycle through the loop.
  if (ierr /= 0) then
   write(*,*) 'Error, invalid input.'
```

```
   cycle interactive_loop
  end if
!   Compute area.  The ** means "raise to a power."
  area = 2 * pi * (radius**2 + radius*height)
!   Write the input variables (radius, height)
!   and output (area) to the screen.

  write (*,'(1x,a7,f6.2,5x,a7,f6.2,5x,a5,f6.2)') &
   'radius=',radius,'height=',height,'area=',area
  yn = ' '
  yn_loop: do
   write(*,*) 'Perform another calculation? y[n]'
   read(*,'(a1)') yn
   if (yn=='y' .or. yn=='Y') exit yn_loop
   if (yn=='n' .or. yn=='N' .or. yn==' ') exit
interactive_loop
  end do yn_loop
 end do interactive_loop
end program cylinder
```

از فرترن ۹۰ توانایی به کارگیری حافظه‌ی پویا به این زبان افزوده شد. نمونه‌ی زیر چگونگی به کارگیری حافظه‌ی پویا در این زبان را نشان می‌دهد.

```
program average
 ! Read in some numbers and take the average
! As written, if there are no data points, an average of
zero is returned
! While this may not be desired behavior, it keeps this
example simple
implicit none
integer :: number_of_points
real, dimension(:), allocatable :: points
real :: average_points=0., positive_average=0.,
negative_average=0.
write (*,*) "Input number of points to average:"
read (*,*) number_of_points
allocate (points(number_of_points))
write (*,*) "Enter the points to average:"
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
read (*,*) points
! Take the average by summing points and dividing by
number_of_points
if (number_of_points > 0) average_points =
sum(points)/number_of_points
! Now form average over positive and negative points
only
 if (count(points > 0.) > 0) positive_average =
sum(points, points > 0.) &
   /count(points > 0.)
 if (count(points < 0.) > 0) negative_average =
sum(points, points < 0.) &
   /count(points < 0.)
 deallocate (points)
! Print result to terminal
 write (*,'(''Average = '', 1g12.4)') average_points
 write (*,'(''Average of positive points = '', 1g12.4)')
positive_average
 write (*,'(''Average of negative points = '', 1g12.4)')
negative_average
end program average
```

مفهوم اشاره‌گرها در زبان فرترن با مفهوم آن در زبان C متفاوت است. اشاره‌گرها در زبان فرترن فقط دربردارنده‌ی آدرس حافظه نیستند بلکه دربردارنده‌ی اطلاعات اضافه‌تری همچون مرتبه‌ی هدف، محدوده‌های بالا و پایین هر بُعد و حتی گام‌های پیمایش آن نیز هستند. به این ترتیب اشاره‌گرهای زبان فرترن ۹۰ می‌توانند به یک زیر ماتریس نیز اشاره کنند. اشاره‌گرها به یک متغیر اشاره می‌کنند یا به حافظه‌ی پویا اشاره می‌کنند و هیچ گونه محاسبات اشاره‌گری در این زبان وجود ندارد. در برنامه‌ی زیر این توانایی زبان فرترن نشان داده شده است.

```
module SomeModule
 implicit none
contains
 elemental function A(x) result(res)
   integer :: res
   integer, intent(IN) :: x
   res = x + 1
 end function
end module SomeModule
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
program Test
 use SomeModule, DoSomething => A
 implicit none
  !Declare variables
 integer, parameter :: m = 3, n = 3
 integer, pointer :: p(:)=>null(), q(:,:)=>null()
 integer, allocatable, target :: A(:,:)
 integer :: istat = 0, i, j
 character(80) :: fmt

!  Write format string for matrices
!  (/ A / A, " = [", 3( "[",3(i2, 1x), "]" / 5x), "]" )
 write (fmt, '("(/ A / A, "" = [""", ", i0, "( ""[""," ,
i0, "(i2, 1x), ""]"" / 5x), ""]"" )")') m, n

 allocate(A(m, n), q(m, n), stat = istat)
 if (istat /= 0) stop 'Error during allocation of A and
q'
!  Matrix A is:
!  A = [[ 1  4  7 ]
!       [ 2  5  8 ]
!       [ 3  6  9 ]
!       ]
 A = reshape([(i, i = 1, size(A))], shape(A))
 q = A
  write(*, fmt) "Matrix A is:", "A", ((A(i, j), j = 1,
size(A, 2)), i = 1, size(A, 1))
!  p will be associated with the first column of A
 p => A(:, 1)
!  This operation on p has a direct effect on matrix A
 p = p ** 2
!  This will end the association between p and the
first column of A
 nullify(p)
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
!  Matrix A becomes:
!  A = [[ 1  4  7 ]
!       [ 4  5  8 ]
!       [ 9  6  9 ]
!       ]
 write(*, fmt) "Matrix A becomes:", "A", ((A(i, j), j =
1, size(A, 2)), i = 1, size(A, 1))
!  Perform some array operation
 q = q + A
!  Matrix q becomes:
!  q = [[ 2  8 14 ]
!       [ 6 10 16 ]
!       [12 12 18 ]
!       ]
 write(*, fmt) "Matrix q becomes:", "q", ((q(i, j), j =
1, size(A, 2)), i = 1, size(A, 1))
!  Use p as an ordinary array
 allocate (p(1:m*n), stat = istat)
 if (istat /= 0) stop 'Error during allocation of p'
!  Perform some array operation
 p = reshape(DoSomething(A + A ** 2), shape(p))
!  Array operation:
!      p(1) = 3
!      p(2) = 21
!      p(3) = 91
!      p(4) = 21
!      p(5) = 31
!      p(6) = 43
!      p(7) = 57
!      p(8) = 73
!      p(9) = 91
 write(*, '("Array operation:" / (4x,"p(",i0,") =
",i0))') (i, p(i), i = 1, size(p))
  deallocate(A, p, q, stat = istat)
 if (istat /= 0) stop 'Error during deallocation'
end program Test
```

دستور case در زبان جالب است.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
 SELECT CASE (number)        ! number of type integer
  CASE (:-1)                  ! all values below 0
   n_sign = -1
  CASE (0)                    ! only 0
   n_sign = 0
  CASE (1:)                   ! all values above 0
   n_sign = 1
  END SELECT
```

و برای حالت پیش‌فرض

```
CASE DEFAULT
```

یکی از توانایی‌هایی دیگری که به این زبان در نسخه‌ی ۲۰۰۸ آن افزوده شد Coarray ها هستند. این‌ها متغیرهای ویژه‌ای هستند که می‌توانند میان چندین نمونه از یک برنامه، به نام image، به اشتراک گذاشته شوند . سودمندی اصلی این‌ها فرایهم کردن یک سطح بالای ارتباط میان فرترن و همروندی است. همچنین توانایی‌های همگام سازی نیز برای آن فراهم شده است. در تعریف این آرایه‌ها به جای پرانتز ، کروشه به کار برده می‌شود.

۲.۲. زبان GO!

این زبان در Google گسترش یافته است و دستور زبان آن به زبان‌های C و python بسیار نزدیک است کوشیده شده است که توانایی‌های این زبان‌ها را با هم داشته باشد. همچنین این زبان یک زبان کامپایلی است که بتوان کد اجرایی را مستقیم روی سخت‌افزار اجرا نماید و سرعت بیشتری داشته باشد.

برنامه‌هایی از این زبان در زیر گذاشته شده است.

```
package main

import (
   "os"
   "flag"  // command line option parser
)

var omitNewline = flag.Bool("n", false, "don't print
final newline")

const (
   Space = " "
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
    Newline = "\n"
)

func main() {
    flag.Parse()    // Scans the arg list and sets up
flags
    var s string
    for i := 0; i < flag.NArg(); i++ {
        if i > 0 {
            s += Space
        }
        s += flag.Arg(i)
    }
    if !*omitNewline {
        s += Newline
    }
    os.Stdout.WriteString(s)
}
```

سایت اصلی این زبان در زیر گذاشته شده است.

http://golang.org/

۳.۲. زبان limbo

این زبان در آزمایشگاه‌های بل (Bell) گسترش یافته است و هدف آن سامانه‌های توزیع شده بـر روی رایانه‌هـای کوچک است. این زبان شباهت فراوانی به Pascal و C++ دارد.

توضیح کاملی از این زبان را دنیس ریچی (یکی از سازندگان زبان C) در پیوند زیر داده است.

http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo

در زیر دستورهایی از این زبان نوشته شده است.

```
 printconst(c: ref Constant)
 {
   sys->print("%s: ", c.name);
   pick x := c {
```

```
    Str =>
      sys->print("%s\n", x.s);
    Pstring =>
      sys->print("[%s]\n", x.s);
    Real =>
     sys->print("%f\n", x.r);
    };
  }
```

```
reader(keys: chan of int, dfd: ref FD)
{
  n: int;
  b:= array[1] of byte;
  for(;;) {
    n = sys->read(dfd, b, 1);
    if(n != 1)
      break
    case int b[0] {
    '0' =>  n = Ir->Zero;
    '1' =>  n = Ir->One;
    . . .
    16r7f =>    n = Ir->Mute;
    * =>    n = Ir->Error;
    }
    keys <-= n;
  }
  keys <-= Ir->Error;
}
```

زبان دیگری که از سوی Bell ساخته شده است Alef نام دارد که در آدرس زیر توضیح هایی دربـاره‌ی آن داده شده است.

http://doc.cat-v.org/plan_9/2nd_edition/papers/alef/ref

فهرستی از این زبان‌ها در زیر گذاشته شده است.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

http://quotes.cat-v.org/programming/

۴.۲. زبان Groovy (programming language)

این زبان بر پایه‌ی زبان python ساخته شده است و هدف آن افزایش توانایی‌های زبان python است. این زبان بر سکوی java اجرا می‌شود و از توانایی‌های زبان java به طور مستقیم بهره می‌برد.

در سایت زیر توضیح‌هایی درباره‌ی این زبان نوشته شده است.

http://en.wikipedia.org/wiki/Groovy_(programming_language)

در ادامه برنامه‌هایی به این زبان نوشته شده است.

```
fullString = ""
orderParts = ["BUY", 200, "Hot Dogs", "1"]
orderParts.each {
 fullString += it + " "
}

println fullString
```

```
myMap = ["China": 1 , "India" : 2, "USA" : 3]

result = 0
myMap.keySet().each( { result+= myMap[it] } )
println result
```

```
myFileDirectory = "C:\\temp\\"
myFileName = "myfile.txt"
myFile = new File(myFileDirectory + myFileName)

printFileLine = { println "File line: " + it }

myFile.eachLine( printFileLine )
```

```
"potatoe" ==~ /potatoe/
```

```
def checkSpelling(spellingAttempt,
spellingRegularExpression)
{
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
   if (spellingAttempt ==~ spellingRegularExpression)
   {
      println("Congratulations, you spelled it
correctly.")
    } else {
      println("Sorry, try again.")
   }
}

theRegularExpression = /Wisniewski/
checkSpelling("Wisniewski", theRegularExpression)
checkSpelling("Wisnewski", theRegularExpression)


theRegularExpression = /Wisn(ie|ei)w?ski/
checkSpelling("Wisniewski", theRegularExpression)
checkSpelling("Wisnieski", theRegularExpression)
checkSpelling("Wisniewewski", theRegularExpression)

[ 1, 2, 3, 4 ].collect(square)

http://groovy.codehaus.org
```

۵.۲. زبان javascript

این زبان در شرکت Netscape طراحی شد و برای اجرای کدهای سمت سرویس گیرنده روی مرورگر (web-browser) آماده شد. سپس این شرکت برای استاندارد سازی این زبان آن را به مؤسسه‌ی استاندارد ECMA سپرد. از این پس نام زبان ECMA با نسخه‌های گوناگون آن نیز به چشم می‌خورد که بر پایه‌ی نسخه‌های استاندارد شده‌ی این زبان است. باگذشت زبان بیشتر مرورگرها استاندارد را پیروی کردند و این زبان یک زبان پرطرفدار در دنیای وب گردید.

از کارهای جالبی که ECMA به تازگی انجام داده است آزمایش‌هایی است که به کمک آن اندازه‌ی سازگاری مرورگرهای گوناگون با استاندارد این زبان سنجیده می‌شود و به صورت عمومی در اختیار همگان قرار می‌گیرد. تقریباً مانند همیشه در میان مرورگرهای معروفی چون IE , safari, chrome , opera , firefox بدترین نتیجه را IE داشته است که میزان سازگاری کمتری با آزمایش‌های ECMA داشته است.

۶.۲. زبان Erlang

این زبان یکی از زبان‌های تابعی است که همزمان کوشیده است دستورهایی از زبان‌های رویه‌ای نیز به همراه خود داشته باشد تا توانایی بهتری داشته باشد. شاید بتوان گفت زبان F# از ایـن زبـان و زبـان Ocaml الهـام گرفتـه است. سایت رسمی این زبان در زیر گذاشته شده است.

http://www.erlang.org/doc.html

در زیر دستورهایی از زبان Erlang نوشته شده است.

```
if
  Condition 1 ->
    Action 1;
  Condition 2 ->
    Action 2;
  Condition 3 ->
    Action 3;
  Condition 4 ->
    Action 4
end
```

```
  Leap = if
    trunc(Year / 400) * 400 == Year ->
      leap;
    trunc(Year / 100) * 100 == Year ->
      not_leap;
    trunc(Year / 4) * 4 == Year ->
      leap;
    true ->
      not_leap
  end,
  case Month of
    sep -> 30;
    apr -> 30;
    jun -> 30;
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
  nov -> 30;
  feb when Leap == leap -> 29;
  feb -> 28;
  jan -> 31;
  mar -> 31;
  may -> 31;
  jul -> 31;
  aug -> 31;
  oct -> 31;
  dec -> 31
end.
```

فصل سوم

زبان‌های آموزشی

۳. زبان‌های آموزشی

در این فصل برخی زبان‌های آموزشی بررسی شده است. این زبان‌ها بیشتر ویـژه‌ی نوآمـوزان و نوجوانـان هسـتند. دسته‌ای از زبان‌ها نیز هستند که برای کودکان نوشته شده‌اند که برخـی از آن‌هـا همـراه بـا ابزارهـای گرافیکـی و بصری توانمند هستند مانند زبان‌های زیر

Alice

http://www.alice.org/

GameKit

http://cs.brown.edu/people/morgan/gamekit/

Phorogram

http://phrogram.com/

Kidslike

http://www.kidslike.info

Kidslike

http://www.kidslike.info

فهرستی از این زبان‌ها در سایت‌های زیر گذاشته شده است.

http://en.wikipedia.org/wiki/Educational_programming_language

http://www.dmoz.org/Computers/Programming/Languages

http://www.thefreecountry.com/compilers/educational-programming-languages.shtml

۱.۳. زبان LOGO

یکی از نخستین زبان‌های ویژه‌ی آموزش برنامه نویسی است که چندین زبان آموزشی دیگر بر پایه‌ی آن طراحی و پیاده‌سازی شد.

۲.۳. زبان kidbasic یا Basic256

این زبان گونه‌ای ساده شده‌ای از زبان بیسیک برای آموزش برنامه نویسی به کودکان است. این زبان ساختارهای سنتی برنامه نویسی در بیسیک مانند goto ، for/next ، gosub و همانند آن را دارد که کمک می‌کند یادگیرنده به سادگی مفهوم‌های کنترل برنامه را یاد بگیرد. این زبان دربردارنده‌ی حالت گرافیکی ساده‌ای است که این امکان را می‌دهد تا به سادگی بتوان شکل‌های گرافیکی را به کمک این زبان کشید. همچنین یک مجموعه‌ی غنی از توضیح‌ها و نمونه‌ها آماده شده است که به سادگی به کمک نمونه‌های ارائه شده یادگیرنده گام به گام با مفهوم‌ها را فرامی‌گیرد. این مجموعه از توضیح‌ها و نمونه‌ها دربردارنده‌ی تمرین‌های جالب برنامه نویسی نیز هست.

آدرس سایت اصلی این زبان در زیر گذاشته شده است.

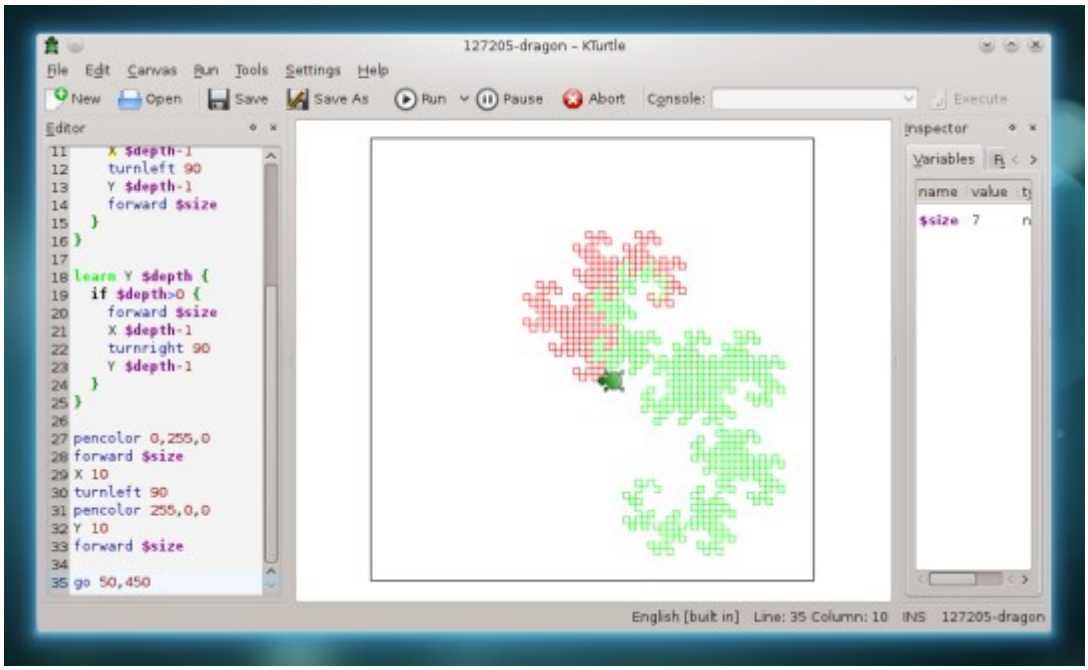
```
http://sourceforge.net/projects/kidbasic/
```

۳.۳. KTurtle

Kturtle یک محیط آموزشی برنامه نویسی بر پایه‌ی زبان TurtleScript است. زبان TurtleScript الهام گرفته از زبان Logo است. کاربران (یا برنامه نویسان) TurtleScript دستورهایی را برای کنترل turtle (لاک پشت) می‌دهند. turtle بر روی یک سطح گرافیکی است که کمک می‌کند به سادگی بتوان مفهوم‌های پایه‌ی ریاضی، هندسی و برنامه نویسی را در آن آموخت.

این بسته (محیط آموزشی) به همراه KDE و در بخش آموزشی آن ارائه می‌شود. آدرس سایت اصلی آموزشی KDE در زیر گذاشته شده است.

```
http://edu.kde.org/
```



شکل ۱.۳: محیط ابزار Kturtle

محیط اجرا دربردارنده‌ی سه بخش است که بخش سمت چپ یک ویرایشگر برای برنامه نویسی است وسط خروجی برنامه را که حرکت لاک پشت است نشان می‌دهد و سمت راست توضیح‌هایی درباره‌ی متغیرها، تابع‌ها و دیگر پارامترها می‌دهد. در زیر برنامه‌ی ساده‌ای ب این زبان نوشته شده است.

```
reset
# this program has been made by Cies Breijs.
canvassize 200,200
canvascolor 0,0,0
pencolor 255,0,0
penwidth 5

go 20,20
direction 135

forward 200
turnleft 135
forward 100
turnleft 135
forward 141
turnleft 135
forward 100
turnleft 45

go 40,100
```

برای ساخت یک دستور تازه دستور learn به کار برده می‌شود.

```
learn faculty $x {
 $r = 1
 for $i = 1 to $x {
  $r = $r * $i
 }
 return $r
}

print faculty 5
```

دستورهای حلقه‌ی این زبان repeat, while و for هستند.

```
$x = 1
while $x < 5 {
 forward 10
 wait 1
 $x = $x + 1
}

while boolean { ... }
repeat number { ... }
for variable = number to number step number { ... }
break
```

دستور شرطی این زبان if است.

```
reset
$x = 4
if $x > 5 {
 print "$x is greater than five!"
} else {
 print "$x is smaller than six!"
}
```

اگر پیش از این با Turbo C یا Borland C و تابع‌های conio.h در آن مانند gotoxy و همانند آن کار کرده باشید خواهید دید که این کتابخانه بر پایه‌ی همان ایده‌ی زبان LOGO و حرکت مکان نما در صفحه آماده شده است و در زمان خود نیز بسیار دوست‌داشتنی و پرکاربرد بود.

GVR .۴.۳

Guido van Robot یک زبان آموزشی برای حرکت یک آدمک در فضای دو بُعدی است. دستورهای این زبان با زبان‌ها متداول تفاوت زیادی دارد زیرا خود را محدود به حرکتِ یک آدمک در صفحه کرده است. نمونه‌هایی از کدهای این زبان در زیر نوشته شده است.

پنج دستور پایه‌ای این زبان عبارت هستند از

```
move
turnleft
pickbeeper
putbeeper
turnoff
```

برای گروه بندی کردن چند دستور:

```
<instruction>
<instruction>
  ...
<instruction>
```

دستور شرط

```
if <test>:
    <block>

if <test>:
    <block>
else:
    <block>

if <test>:
    <block>
elif <test>:
    <block>
...
elif <test>:
    <block>
else:
    <block>
```

تعدادی از شرط‌ها:

```
front_is_clear
front_is_blocked
left_is_clear
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
left_is_blocked
right_is_clear
right_is_blocked

next_to_a_beeper
not_next_to_a_beeper
any_beepers_in_beeper_bag
no_beepers_in_beeper_bag

facing_north
not_facing_north
facing_south
not_facing_south
facing_east
not_facing_east
facing_west
not_facing_west
```

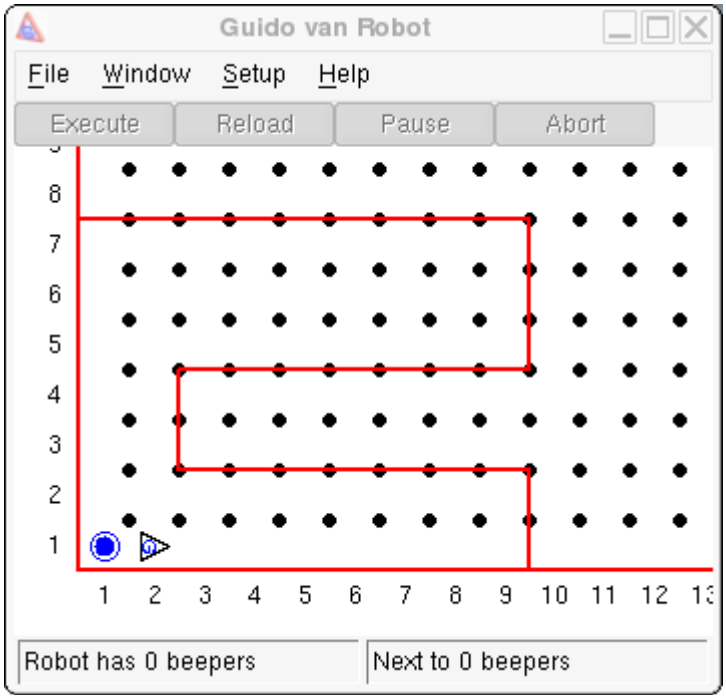
```
do <positive_number>: حلقه
  <block>
while <test>: حلقه
  <block>
define <new_name>: تعریف دستور جدید
  <block>
turnoff    پایان برنامه
```

یک برنامه نمونه برای حرکت در صفحه :

```
define turnright:
  do 3:
    turnleft
define follow_right_wall:
  if right_is_clear:
    turnright
    move
  elif front_is_clear:
    move
  else:
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
    turnleft
while not_next_to_a_beeper:
  follow_right_wall
turnoff
```



شکل ۲.۳: خروجی برنامه‌ی نمونه برای حرکت در صفحه

آدرس سایت اصلی این زبان در زیر گذاشته شده است.

http://gvr.sourceforge.net/

littlewizard .۵.۳

یک ابزار آموزشی برای کودکان است که به صورت گام به گام با مفهوم‌های گوناگون ریاضی و رایانه آشنا شـوند و به کمک بخش‌های گوناگون گرافیکی می‌کوشد مفهوم‌ها را به کودکان بیاموزد.

آدرس سایت اصلی این زبان در زیر گذاشته شده است.

http://littlewizard.sourceforge.net/

Small Basic .۶.۳

زبانی آموزشی از شرکت microsft بر پایه‌ی زبان Bsic اسـت. از ویژگی‌هـای ایـن زبـان تعـداد کـم کلمه‌هـای کلیدی آن و به کارگیری شیء گرایی است.

متغیرهای این زبان مانند دیگر زبان‌ها هستند با این تفاوت که طول آن‌ها محدود به چهل نویسه است. نوع‌های عددی (صحیح و اعشار) ، رشته‌ای و منطقی (Boolean) در این زبان وجود دارد. نمایه‌ی آرایه‌ها در این زبان از صفر آغاز می‌شوند.

نمونه‌هایی از کدهای به این زبان در ادامه گذاشته شده است.

```
StartTime = Now
ExplorerName = "Captain Spaulding"
TextWindow.Title = "My Program"
BitCount = ByteCount * 8
Energy = Mass * LightSpeed * LightSpeed
NetWorth = Assets - Liabilities
```

```
If (Balance - Check < 0) Then
 Trouble = "true"
 CheckingStatus = "Red"
ElseIf (Balance - Check = 0) Then
 Trouble = "false"
 CheckingStatus = "Yellow"
Else
 Trouble = "false"
 CheckingStatus = "Black"
EndIf
```

```
Rolls = 0
Counter = 0
While (Counter < 10)
 '  Roll a simulated die
 Rolls = Rolls + 1
 If (Math.GetRandomNumber(6) = 6) Then
  Counter = Counter + 1
 EndIf
EndWhile
```

```
Sum = 0
Rolls = 0
SumLoop:
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
 ' Roll a simulated die
 Die = Math.GetRandomNumber(6)
 Sum = Sum + Die
 Rolls = Rolls + 1
If (Sum <= 30) Then
 Goto SumLoop
EndIf
```

```
' Guide to Small Basic, Example 3-3
TextWindow.Title = "Example 3-3"
TextWindow.WriteLine("Line 1")
Program.Delay(1000)
TextWindow.WriteLine("Line 2")
Program.Delay(2000)
Program.End()
```

```
' Guide to Small Basic, Example 4-3
TextWindow.Title = "Example 4-3"
TextWindow.BackgroundColor = "White"
TextWindow.ForegroundColor = "Black"
TextWindow.Clear()
GetNumberSixes:
TextWindow.Write("How many sixes must be rolled? ")
Number = TextWindow.ReadNumber()
Rolls = 0
Counter = 0
While (Counter < Number)
 '  Roll a simulated die
 Rolls = Rolls + 1
 If (Math.GetRandomNumber(6) = 6) Then
  Counter = Counter + 1
 EndIf
EndWhile
TextWindow.WriteLine("It took " + Rolls + " rolls to get
" + Number + " sixes.")
TextWindow.WriteLine("")
Goto GetNumberSixes
```

آدرس سایت اصلی این زبان و همچنین راهنمای آن در زیر گذاشته شده است.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

http://msdn.microsoft.com/en-us/beginner/hh304480.aspx
http://blogs.msdn.com/b/smallbasic/

همچنین آدرس زیر برای نو آموزان برنامه نویسی است.

http://msdn.microsoft.com/en-us/beginner/default.aspx

ROBO .۷.۳

یک زبان آموزشی بسیار ساده به همراه محیط توسعه‌ی کاربر پسند است که کمک می‌کند تا نو آموزان رایانه مفهوم‌های پایه‌ی علم رایانه را بسادگی بیاموزند. این زبان دستورهایی را به یک آدمک می‌دهد تا اجرا کند. در زیر تعدادی از دستورهای این زبان نوشته شده است. یک آدمک کارهای همچون دیدن(see)، رنگ کردن(paint)، حرکت کردن(move)، گرفتن شیء(grab) و پرتاب سکه (برای انتخاب تصادفی flip coin) را دارد.

move

forward(n)	Move n steps forward
backward(n)	Move n steps backward
left()	Turn left over 90 degrees
right()	Turn right over 90 degrees
north(n)	Turn to head north and move n steps forward
south(n)	Turn to head south and move n steps forward
east(n)	Turn to head east and move n steps forward
west(n)	Turn to head west and move n steps forward

Paint

paintWhite()	Put the brush with white paint to the ground.
paintBlack()	Put the brush with black paint to the ground.
stopPainting()	Stop painting, hide the brush

Grab

pickUp()	Get the beacon in front of the robot
putDown()	Put a beacon in front of the robot

Flip coin

coinFlip() Flip a coin to make a random choice. flipCoin() will either be true or false with a chance of 50%-50%.

See

Left	Front	Right
leftIsObstacle()	frontIsObstacle()	rightIsObstacle()
leftIsClear()	frontIsClear()	rightIsClear()
leftIsBeacon()	frontIsBeacon()	rightIsBeacon()
leftIsWhite()	frontIsWhite()	rightIsWhite()
leftIsBlack()	frontIsBlack()	rightIsBlack()

سایت اصلی این زبان و محیط توسعه در زیر گذاشته شده است.

http://robomind.net/en/index.html

فصل چهارم

زبان‌های برنامه نویسی فارسی

پیشین

۴. زبان‌های برنامه نویسی فارسی پیشین

تا کنون کوشش‌هایی برای ساخت زبان برنامه نویسی فارسی انجام شده است که به خوبی نیز اطلاع رسانی نشده است. ایده‌ی ساخت این زبان در ذهن بسیاری از برنامه نویسان بوده است و گاه نیز توسعه‌های کوچکی بر روی آن انجام داده‌اند. در این فصل کوشیده شده است چند زبانی را که پیش از این ساخته شده است معرفی شود. این فهرست زبان‌ها با و جود بسیار زیادی که انجام شد مطلقاً کامل نیست و باید توسعه دهندگان دیگری نیز زبان‌های دیگری را آماده کرده باشند.

گفته می‌شود که حتی برای رایانه‌های comodore 64 نیز برنامه‌ای نوشته شده بود که برنامه‌ی شبه بیسیک به زبان فارسی را گرفته و به بیسیک تبدیل می‌کرده است. مشکل دیگری نیز که اغلب در ایران دیده می‌شود رعایت نکردن کپی رایت و دزدیدن ایده‌های دیگران است که متأسفانه به سادگی انجام می‌شود و همین باعث می‌شود که کارهای بزرگی چون ساخت یک زبان جدید چندان جدی گرفته نشود. همچنین با توجه به جو جامعه و کوچک انگاری‌های کارهای ایرانی و سرکوفت‌ها و مقایسه‌ی بی‌جایی که می‌شود بسیاری در همان آغاز راه از این راه بازمی‌گردند. نگارنده‌ی خاطره‌ی نه چندان خوشایندی از مقایسه‌ی کارش با یکی از بزرگترین نرم‌افزارهای دنیای کنونی رایانه دارد که جناب داور گرامی کار نگارنده را با آن مقایسه می‌فرمودند و بدین سان به جای انجام کار عملی و کاربردی همه به سوی انجام کارهای نظری و کوتاه مدت سوق داده می‌شوند که نتیجه‌ی مالی نیز داشته باشد و به گونه‌ای نیز به خارج کشور متصل باشد مانند مقاله‌ی خارجی یا جشنواره‌ی خارجی یا همانند.

۱.۴. زبان فارسی دانش

غلامرضا دانشمند این زبان را به عنوان یک پروژه‌ی برنامه نویسی چندین سال پیش انجام داده است. البته کد منبع این ابزار پیاده‌سازی شده در اختیار گذاشته نشده است و همچنین این ابزار دارای گزارش یا توضیح کاملی نیست که بتوان از روی آن توضیح بیشتری داد. آنچه در اختیار دیگران گذاشته شده است یک برنامه اجرایی در ویندوز است. اینکه در چه تاریخی نیز او این زبان را آماده کرده است روشن نیست.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

۲.۴. فارسی نت

این زبان تغییر یافته‌ی زبان Csharp به کلمه‌های فارسی است به گونه‌ای که بتوان به طور کاملاً فارسی با آن برنامه نوشت خروجی آن کد میانی Dot Net است. این ابزار به کمک visual studio کار می‌کند و از ویرایشگرهای آن سود می‌برد. بخش زیر از توضیح نویسنده‌ی این زبان در زیر آورده شده است : http://sites.google.com/site/mrtofigh2/farsinet

برپایه‌ی گفته‌ی سازنده‌ی این زبان، فارسی‌نت حاصل ماه‌ها مطالعه، طراحی و برنامه‌نویسی است. سی‌شارپ، دلفی و اسکویک (اسمالتاک) بر طراحی این زبان موثر بوده‌اند، اما حقیقت این است که فارسی‌نت یک زبان کاملا فارسی (پارسی) است.

اولین برنامه: برنامه زیر واژه سلام را می‌نویسد:

عملگرها

فارسی‌نت دو دسته عملگر دارد؛ عملگرهای ریاضی (با شرکت‌پذیری معمول در ریاضیات) و عملگرهای فارسی (با شرکت‌پذیری راست‌به‌چپ).

```
آ: حقیقی = ۳.۱۴ + شعاع × ۲؛    -- بخوانید: ۲ ضربدر شعاع بعلاوه ۳.۱۴
ب: حقیقی = ۲ در شعاع با ۳.۱۴؛    -- از راست به چپ بخوانید
```

دستورها

```
پیوست فارسی؛
برنامه {
سرآغاز
یاده (ب: صحیح)
}
برای آ: صحیح = ۲ تا ۱۰۰ {
ب = آ بر ۲؛
تا درست مر
اگر ب برابر ۱ {پایانه.بنویس(آ با « »)؛ بیرون}
وگر آ مانده ب برابر ۰ مر بیرون
وگرنه ب -= ۱
}؛
پایانه.انتظار()
{
{
-- خروجی: ۲ ۳ ۵ ۷ ۱۱ ۱۳ ۱۷ ۱۹ ۲۳ ۲۹ ۳۱ ۳۷ ۴۱ ۴۳ ۴۷ ۵۳ ۵۹ ۶۱ ۶۷ ۷۱ ۷۳ ۷۹ ۸۳ ۸۹ ۹۷
```

دستورها با نقطه‌ویرگول جداسازی می‌شوند. «مر» بیان و دستور را جدا می‌کند. دنباله دستورات درون آکولاد قرار می‌گیرد.

شی‌گرایی

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

مدل شیئی فارسی‌نت شبیه سی‌شارپ است. در برنامه زیر رده (کلاس) دانشجو از رده شخص ارث می‌برد.

برای دسترسی به این زبان پیوندهای زیر وجود دارد:

http://sites.google.com/site/mrtofigh2/farsinet
http://groups.google.com/group/farsinet
http://sites.google.com/site/mrtofigh2/
http://mrtofigh.wordpress.com/

گفتگو درباره‌ی این زبان در سایت زیر گذاشته شده است. که در آن میان سازنده‌ی این زبان و برخی از کاربران سایت مشاجره‌های گوناگونی انجام شده است که برخی از آن‌ها اشکال‌ها جدی این زبان هستند و برخلاف ادعای سازنده‌ی آن، این زبان بسیار دشوار بود و هیچ چیز تازه‌ای را در برندارد.

http://barnamenevis.org/forum/showthread.php?t=129487

۳.۴. زبان‌های برنامه نویسی غیر انگلیسی

زبان‌های برنامه نویسی غیر انگلیسی گوناگونی تا کنون ساخته شده است ولی چندان مهم نیستند و چندان هم به کار گرفته نشده‌اند.

http://en.wikipedia.org/wiki/Non-English-based_programming_languages

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

در فهرست داده شده در این صفحه زبان‌های برنامه نویسی گوناگونی دیده می‌شود. همچنین زبان‌های برنامه‌نویسی که این توانایی را دارند که بتوان به زبان‌های گوناگون برای آن‌ها برنامه نویسی کرد و این توانایی را دارند که بتوان زبان را عوض کرد. فهرست آن‌ها در زیر آورده شده است.

Babylscript – A multilingual version of JavaScript which uses multiple tokenizers to support localized keywords in different languages and which allows objects and functions to have different names in different languages

ChinesePython – A complete translation of the Python scripting language into Chinese

Component Pascal – A preprocessor that translates native-language keywords into English in an educational version of the BlackBox Component Builder available as open source from http://www.inr.ac.ru/~info21/software.htm
The translation is controlled via a modifiable vocabulary and supported by modifiable compiler error messages. A complete Russian version is used in education, and it should be possible to accommodate other left-to-right languages (e.g. the Kabardian language has been tried as a proof of concept).

HyperTalk – A programming language, which allows translation via custom resources, used in Apple's HyperCard

Macintosh AppleScript – A language once allowed for different "dialects" including French and Japanese; however, these were removed in later versions
Maude - Completely user definable syntax and semantics, within the bounds of the ASCII character set[4]

Perl – While Perl's keywords and function names are generally in English, it allows modification of its parser to modify the input language, such as in Damian Conway's Lingua::Romana::Perligata module, which allows programs to be written in Latin or his Lingua::tlhInganHol::yIghun Perl language in Klingon.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
They do not just change the keywords but also the
grammar to match the language.
```

```
Protium - A language designed to support any possible
human language
```

یادآوری می‌شود در بسیاری از نسخه‌های جدید زبان‌های برنامه نویسی مانند C و C++ می‌توان نام متغیرها و تابع‌ها را به هر زبانی نوشت و نیازی نیست که این نام‌ها به انگلیسی باشند ولی با این همه این توانایی این زبان‌ها چندان به کار برده نمی‌شود.

۱.۳.۴. زبان برنامه نویسی به عبری

دستورهای این زبان به عبری نوشته می‌شوند و سپس به زبان PHP تبدیل می‌شوند تا روی موتور آن اجرا شوند و یک زبان مستقل نیست. این زبان و محیط اجرای آن مدتها است که به روز نشده است.

```
http://hpl.sourceforge.net/
```

۲.۳.۴. زبان برنامه نویسی کلمات به عربی

این زبان در دو سال گذشته گسترش یافته است و همراه با کوشش فراوانی بوده است تا این زبان افزون بر عربی بودن دارای ویژگی‌های تازه و سودمندی باشد و فقط دربردارنده‌ی عربی شده‌ی یک زبان برنامه نویسی پیشین نباشد. دستور این زبان به خوبی آماده شده است و دربردارنده‌ی نکته‌های جالبی است که البته بیشتر توضیح‌های این زبان به عربی است. یک ویرایشگر خوب و توانمند نیز برای آن آماده شده است تا برنامه نویس بتواند به سادگی برنامه‌ی خود را بنویسد و نگران چپ به راست یا راست به چپ بودن نباشد. همچنین مانند زبان‌های نوینی همچون java و Csharp در این مفسر آماده شده در این زبان در آغاز برنامه به یک زبان میانی تبدیل می‌شود و سپس بر روی یک موتور اجرا این زبان میانی اجرا می‌شود. برای ساختن این زبان ابزارهای متن باز به کار گرفته شده است و بر روی سیستم عامل‌های لینوکس، ویندوز و مکینتاش به سادگی اجرا می‌شود این قابل حمل بودن بسیار برای این ابزار سودمند است.

سایت‌های مربوط به این زبان در زیر گذاشته شده است.

```
http://www.kalimat-lang.com
http://code.google.com/p/kalimat/
http://iamsamy.blogspot.com/
http://iamsamy.blogspot.com/2011/06/blog-post.html
```

در زیر برنامه‌هایی به این زبان نوشته شده است.

```
اطبع ۱۸
اطبع ۱۲ + ۱۳
```

```
اطبع "مرحبا"
اطبع "۵+۵"، ۵+۵
```

```
اطبع "ادخل اسمك"
اقرأ س۱
اطبع "ادخل اسم ابيك"
اقرأ س۲
اطبع "تشرفنا يا"، " "، س۱، " "، س۲
```

```
س = ۱
س = س + ۵
اطبع س
```

```
اقرأ "ادخل العدد الأول "، #أ
اقرأ "ادخل العدد الثاني "، #ب

إذا أ>ب :
    اطبع "العدد الأول هو الأكبر"
وإلا إذا ب>أ :
    اطبع "العدد الثاني هو الأكبر"
وإلا:
    اطبع "العددان متساويان"
تم
```

```
لكل أ من ۱ إلى ۵:
    اطبع أ
تابع
```

```
أ=۱
علامة س
اطبع أ
أ = أ + ۱
إذا أ<=۵: اذهب إلى س
```

```
اقرأ "كم عددا تريد أن تجمع؟"، #ع
م = مصفوفة(ع)
لكل أ من ۱ إلى ع:
    اقرأ "ادخل قيمة:"، م[أ]
تابع
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
ج = ۰
م = [۵، ۳۸، ۱۲، ۷۰، ۱۲۰، ۸، ۵، ۳]
لکل أ من ۱ إلی عدد(م):
ج = ج + م[أ]
تابع
اطبع "المجموع هو "، ج
```

فصل پنجم

ویژگی‌های پایه‌ای زبان جدید

۵. ویژگی‌های پایه‌ای زبان جدید

در این فصل برخی از ویژگی‌های پایه‌ای زبان جدید توضیح داده شود. کوشش شده است که مفهوم‌های برنامه نویسی به شکل ساده‌تری در این زبان نوشته شوند تا یادگیری برنامه نویسی ساده‌تر گردد.

۱.۵. متغیرها

متغیرها در این زبان برنامه نویسی چند تفاوت بزرگ با دیگر زبان‌های برنامه نویسی دارند.

الف_ متغیرهای این زبان می‌توانند به هر زبانی نوشته شوند. به طور پیش‌فرض پرونده‌های برنامه‌های نوشته شده به این زبان utf-8 هستند بنابراین می‌توان به سادگی متغیرهای آن را به هر زبانی نوشت. در نخستین گام فقط نویسه‌های زبان‌های انگلیسی و فارسی پذیرفته می‌شود.

ب_ در میان بخش‌های یک متغیر می‌تواند فاصله (space) گذاشته شود. این ویژگی بسیار برای ساده‌تر شدن کار برنامه نویس فارسی مهم است زیرا حروف کوچک و بزرگ در زبان فارسی معنایی ندارند و گذاشتن خط فاصله نیز به هم ریختگی‌هایی را ایجاد می‌کند. بنابراین گذاشتن فاصله بسیار کارساز خواهد بود و کار برنامه نویس را ساده‌تر می‌کند. تنها چیزی که باید دقت شود این است که مفسر این زبان به طور خودکار همه‌ی فاصله‌ها و نیم فاصله‌ها (zero width non-joiner) برداشته می‌شوند و به جای هر کدام از آن‌ها یک فاصله می‌گذارد. در به کارگیری فاصله باید دقت نمود تا مشکلی پیش نیاید و متغیرهای یکسان نام‌های گوناگونی به دلیل بود یا نبود فاصله در میان آن‌ها به وجود نیاید.

دیگر نویسه‌هایی که در دیگر زبان‌های برنامه نویسی برای نام متغیرها به کار برده می‌شود مانند ـ و عددها (پس از نخستین نویسه) می‌تواند در نام متغیرها به کار برده شود.

۲.۵. عددها

در این زبان عددهای ثابت می‌توانند به سادگی نوشته شوند عددهای ثابت دو نوع صحیح و اعشاری دارند که در این نسخه از زبان به ترتیب برابر long long int و long double از زبان C هستند و نوع‌های کوچکتر فقط برای نوع صحیح کاربرد دارد که در نسخه‌های آینده تعیین نوع صریح نیز به این زبان افزوده خواهد شد مانند

آنچه در زبانی مانند C وجود دارد.

```
۱۳۳۴۳
۳۴۲۳۵۳۵۳۵
۰
۲
۲/۰
۲/۴
۰/۲
```

دقت شود که پیش از ممیز و پس از آن باید عدد گذاشته شود زیرا نقطه در این زبان در جاهای گوناگون کاربردهای گوناگونی دارد. برای توان عدد ثابت نیز همان علامت توان به کار برده می‌شود. در اینجا توان‌ها و دیگر عملگرها از راست به چپ هستند بنابراین در خط نخست ۳ به توان ۴ رسیده است. در زبان انگلیسی ممیز با همان نقطه نشان داده می‌شود ولی در فارسی / به کار برده می‌شود که در انگلیسی تقسیم است.

```
۳^۴
۳^۲
۲^۸
```

عمل‌های روی عدد ثابت در زمان کامپایل انجام می‌شود و عدد نتیجه جایگزین آن‌ها می‌شود.

```
۳^۱۰*۴
۴^۱۰ * ۰/۱۴۸۶۷۵
```

دو عبارت بالا و دو عبارت پایین یکسان هستند. فقط پس از عدد نباید فاصله گذاشته شود و پشت سر آن «توان» نوشته شود.

```
۴تو ان ۳
۴ توان ۰/۱۴۸۶۷۵
```

مشکل بزرگی که هنوز به چشم می‌آید نوشتن شدن عددها و عبارت‌های محاسباتی از چپ به راست است و عادت به اینکه عبارت‌های ریاضی را که درون آن‌ها فقط عدد است از چپ به راست بخوانیم. در حالی که نوشته‌های فارسی از راست به چپ هستند و برای برنامه نویسی سردرگمی به بار می‌آورد. اینکه چگونه باید این مشکل حل شود و بهترین راه حل چیست هنوز پاسخی برای آن نیافته‌ام.

۳.۵. رشته‌های ثابت

رشته‌های ثابت با گیومه (در انگلیسی "constant string" به کمک double quotation و در فارسی «رشته‌ی ثابت») نشان داده می‌شوند. علامت quotation معمولی ' اکنون کاربردی ندارد. تصمیم گیری درباره‌ی علامت‌هایی همچون n\ و t\ دشوار است زیرا حرف یا کلمه‌های فارسی جایگزین آن‌ها شود و همچنین به دلیل از راست به چپ بودن باید میان \ و / تفاوت گذاشت.

برای رشته‌های ثابت چند خطی @:@ به همراه یک کلمه چسبیده به آن به کار برده می‌شود و پس از

نویسه‌ی خط جدید رشته‌ی ثابت چند خطی است آغاز می‌شود. پایان رشته ثابت چند خطی نیز همان کلمه می‌آید که پس از آن @:@ و یک فاصله یا خط جدید گذاشته می‌شود.

```
رشته ۱ = @:@رشته‌ثابت
هر چیزی می‌تواند
اینجا نوشته شود.
رشته‌ثابت@:@
```

اگر پس از @:@ علامتی نیاید آن‌گاه همین پایان رشته خواهد بود.

```
رشته ۲ = @:@
هر چیزی می‌تواند
اینجا نوشته شود.
@:@
```

۴.۵. توضیح‌ها

توضیح‌های یک خطی با @ آغاز می‌شوند و توضیح‌های چند خطی درون @{ و }@ گذاشته می‌شوند. اگر بخواهیم در نخستین نویسه‌ی توضیح یک خطی @ را بگذاریم آن‌گاه باید یک فاصله پس از نخستین @ بگذاریم تا با توضیح چند خطی اشتباه نشود. @ { توضیح یک خطی است که نخستین نویسه‌ی توضیح آن { است. همچنین نباید @:@ در آغاز توضیح چند خطی به کار برده شود زیرا @:@ کاربرد دیگری دارد.

۵.۵. کلمه‌های کلیدی

همه‌ی کلمه‌های کلیدی این زبان با نقطه آغاز می‌شوند تا از شناسه‌ها جدا شوند. فهرست کلمه‌های کلیدی این زبان در زیر نوشته شده است. برخی از این کلمه‌های کلیدی برای آینده در نظر گرفته شده‌اند.

کلمه کلیدی	Keyword	کلمه کلیدی	keyword
.یااگر	.elif	.اگر	.if
.تاهنگامیکه	.while	.وگرنه	.else
.تعریف	.def	.چرخه	.for
.هنگامیکه	.when	.گزینش	.select
.بشکن	.break	.بروبه	.goto
.بگذار	.set	.بگیر	.get

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

keyword	کلمه کلیدی	Keyword	کلمه کلیدی
.return	.برگرد	.value	.مقدار
.struct	.ساختار	.result	.نتیجه
.private	.خصوصی	.public	.عمومی
.internal	.داخلی	.protected	.حفاظت
.none	.هیچ	.null	.پوچ
.false	.نادرست	.true	.درست
.long	.بلند	.int	.صحیح
.char	.نویسه	.double	.اعشار
.continue	.ادامه	.string	.رشته
.function	.تابع	.type	.نوع

کلمه‌های کلیدی دیگری نیز به این زبان افزوده خواهد شد که با گسترش زبان نام برای آن‌ها گذاشته می‌شود.

۶.۵. عملگرها

در این زبان تعداد زیادی عملگر وجود دارد و افزایش و گوناگونی عملگرها یکی از هدف‌های این زبان است زیرا به بیشتر بودن عملگرها به سادگی برنامه کمک می‌کند.

۱.۶.۵. عملگرهای محاسباتی

عملگرهای جمع + ، ضرب * ، تفریق - ، تقسیم / (در انگلیسی) یا ÷ (در فارسی)، باقیمانده % (فقط برای عددهای صحیح)، توان ^ ، منفی – (منهای یگانی)، مانند python نسخه‌ی سوم به بالا هستند و همان کاربرد را دارند. البته ضرب رشته‌ها و عددها در این زبان حذف شده است و خطا گرفته می‌شود. عملگر تقسیم هرگاه از راست به چپ باشد ÷ است و هرگاه از چپ به راست باشد / است. مانند نسخه‌ی ۳ از زبان python نتیجه‌ی تقسیم دو عدد صحیح اگر یک عدد اعشاری می‌شد نتیجه اعشاری خواهد بود و نه اینکه بخش اعشار آن حذف شود.

اولویت عملگرها در این زبان مانند دیگر زبان‌ها متداول است و فقط از راست به چپ خواهد بود. هنگامی که کد برنامه انگلیسی باشد آن‌گاه چپ به راست خواهد بود. اینکه چه زمان برنامه به زبان‌هایی است که از راست به

چپ هستند و چه زمان زبان‌هایی در آن هست کـه از چـپ بـه راسـت اسـت بـه آینـده واگـذار می‌شـود، اکنـون پیش‌فرض همان راست به چپ و زبان فارسی است.

۲.۶.۵. عدد مختلط

نوع عدد مختلط مانند python است و به سادگی می‌توان در این زبان بـا عـددها مختلـط کـار کـرد. روش نوشتن آن‌ها به قرار زیر است.

```
2+4i
2+4j
```

تابع ۱ (۲+i ۴) می‌شود. البته هنوز این بخش در زبان پیاده‌سازی نشده است و شاید نیاز به ویرایش داشته باشد.

۳.۶.۵. عملگرهای مقایسه‌ای

این عملگرها نیز همانند همتایان خود در زبان برنامه نویسی C و php هستند و تنها تفـاوت آن‌هـا از راسـت بـه چپ بودن آن‌ها است.

=== ، =! ، == ، =< ، => ، > ، <

نتیجه‌ای که این عملگرها برمی‌گردانند .درست (true.) یا .نادرست (false.) است و بر خلاف زبـان C صـفر نادرست و غیر صفر درست نیست. === برای بررسی تساوی مقدار و نوع به کار برده می‌شود.

۴.۶.۵. عملگرهای منطقی

مانند زبان C این عملگرها عبارت هستند از && , ll , ! که به همان معنا نیز بـه کـار بـرده می‌شـوند. همچنیـن ثابت‌های .درست و .نادرست نیز برای مقدار منطقـی بـه کـار بـرده می‌شـود. در آینـده شـاید .ناشـناخته نیـز بـه مقدارهای منطقی افزوده شود و شاید در ادامه منطق فازی نیز بتوان به آن افزود ولی اکنون با همـان توانایی‌هـای دیگر زبان‌ها با آن کار انجام می‌شود.

۷.۵. انتساب

انتساب در این زبان مانند دیگر زبان‌ها = است و عملگرهای ترکیبی مانند =+ ، =- ، =* ، =٪ ، =^ ، =\ ، =&& ، =ll نیز کارکردی همانند زبان python دارند که دو عملگر =&& و =ll معنای زیر را دارند

```
الـف &&= ب
الـف = الـف && ب
```

```
الـف ||= ب
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
الف = الف || ب
```

۸.۵. عملگر گسترش یافته‌ی : ؟

همانند این زبان C این عملگر می‌تواند مانند شرط عمل کند که دو حالت درست یا نادرست دارد و مقداری را بر می‌گرداند.

```
عبارت منطقی ؟ عبارت :: عبارت
Logical_expression ? Expression  :: Expression
```

حالت دیگری از این عملگر یک عبارت محاسباتی را در شرط خود دارد که یک عدد صحیح بر می‌گرداند که این عدد صحیح میان ۱ تا تعداد : به اضافه ۱ است.

```
Computational_expression ? Expression :: Expression ::
Expression :: Expression :: Expression :: Expression
```

وابسته به اینکه چه عدد طبیعی‌ای در بخش شرط محاسبه شود یکی از عبارت‌ها اجرا می‌شود. اگر نتیجه‌ی عبارت محاسباتی ۱ بود آن‌گاه عبارت یکم، اگر نتیجه‌ی عبارت محاسباتی ۲ بود عبارت دوم، اگر نتیجه عبارت محاسباتی ۳ بود عبارت سوم و تا آخر اجرا می‌شود و نتیجه را برمی‌گرداند. اگر نتیجه‌ی محاسبه صفر یا منفی بود یا از تعداد : ها بیشتر بود آن‌گاه یک خطا رخ داده است که در بخش استثناها توضیح داده می‌شود.

حالت سوم این عملگر بدون :: است. سمت راست ؟ باید یک آرایه از تابع‌ها گذاشته شود و با توجه به عددی که محاسبه می‌شود تابعی از تابع‌های آرایه محاسبه می‌شود.

```
Computational_expression ? Array_of_functions
```

اگر در زمان اجرا سمت راست آرایه‌ای از تابع‌ها نبود یک استثناء رخ داده است.

۹.۵. تابع بخوان

این تابع برای خواندن از ورودی به کار برده می‌شود و به دو شکل می‌توان آن را به کار برد. در نخستین حالت تابع بخوان مقداری را ورودی می‌خواند و برمی‌گرداند. همچنین می‌توان رشته‌ای را درون آن گذاشت تا نمایش دهد. فاصله به عنوان جدا کننده برای این دستور به کار برده می‌شود.

```
متغیر ۱ = بخوان()
متغیر ۱ = بخوان(«یک عدد وارد کنید »)
```

حالت پیشرفته‌تری از این دستور که در آینده بر روی آن بیشتر کار خواهد شد حالتی است که این تابع بتواند یک‌جا چندین مقدار را بخواند و به صورت سطر و ستونی این کار انجام دهد. برای نمونه تابع بخوان زیر یک ماتریس ۶ در ۴ را از ورودی می‌گیرد که در هر سطر ۶ عنصر و در هر ستون ۴ عنصر قرار دارد میان سطرها نویسه‌ی خط بعدی و میان ستون‌ها فاصله گذاشته شده است.

```
متغیر ۲ = بخوان(«»؛۴؛۶)
```

اگر بخواهیم یک خط را تا آخر بخواند و به فاصله‌ها توجه نکند تابع بخوان به صورت زیر به کار برده می‌شود.

```
متغیر ۲ = بخوان (۱)
```

همچنین در این حالت می‌توان حداکثر تعداد نویسه‌ای که باید خوانده شود را مشخص کرد. در این صورت فقط به همان تعداد خواهد خواند.

```
متغیر ۲ = بخوان (۳)
```

۱۰.۵. تابع بنویس

این دستور برای نوشتن یک خط در خروجی صفحه نمایش به کار برده می‌شود. این دستور می‌تواند چندین آرگومان بگیرد یا هیچ آرگومانی نگیرد و فقط یک خط خالی را نمایش دهد.

```
بنویس ()
بنویس (۲ ؛ ۳ ؛ ۴ ؛ «یک رشته‌ی ثابت»)
```

۱۱.۵. دستور اگر

این دستور مانند بسیاری از زبان‌های دیگر است و مانند python دستور واگر (elif) را نیز دارد که کار برنامه نویسی را ساده‌تر می‌کند. همچنین این دستور نیز مانند زبان‌های دیگر دستور وگرنه را نیز دارد.

```
.اگر تعداد خانه‌ها == ۱۲
    تعداد +=۱
.یا اگر تعداد خانه‌ها == ۱۶
    تعداد +=۲
.وگرنه تعداد +=۳
```

فقط جلوی دستور .وگرنه می‌توان دستور بدنه‌ی آن را گذاشت. برای دستورهای .اگر و .یااگر باید حتماً به خط بعد برود یا اینکه آکولاد باز و بسته شود.

۱۲.۵. دستور برای هر

دستور برای هر در این زبان توانایی‌های بسیاری از زبان‌های دیگر را در بردارد و بر چند گونه است.

۱.۱۲.۵. مانند زبان C

در این حالت مانند زبان c است.

```
.چرخه تعداد=۰ ؛ تعداد < ۵ ؛ تعداد++
    بنویس (تعداد)
```

۲.۱۲.۵. مانند زبان python

در این زبان می‌توان روی یک فهرست پیمایش را انجام داد.

```
فهرست۱= [۱ ؛ ۲ ؛ ۳ ؛ ۴ ]
.چرخه متغیری .در فهرست۱
    بنویس(متغیری)
```

۳.۱۲.۵. مانند زبان fortran

در این حالت این دستور می‌تواند یک عدد یا یک عبارت محاسباتی را بگیرد و نتیجه را یک بار در آغاز حلقه به دست آورد سپس اگر تعداد بیشتر از صفر بود به تعداد آن بدنه‌ی حلقه را اجرا کند.

```
تعداد = ۱
.چرخه ۱۲ {
    بنویس(تعداد)
    تعداد +=  ۱
        {
تعداد =۱
ط=۱۶
.چرخه ط {
    بنویس(تعداد)
    تعداد +=  ۱
        {
```

۱۳.۵. دستور تاهنگامی‌که

این دستور مانند دستورهای همانند خود در دیگر زبان‌ها برای حلقه است.

```
.تاهنگامیکه متغیر۱ < ۱۲
    متغیر۱ +=  ۱ ؛ عدد۲ *= متغیر۱
```

۱۴.۵. دستور گزینش

این دستور مانند دستور select یا switch در دیگر زبان‌های برنامه نویسی است. که البته نیازی به آکولاد باز و بسته ندارد.

```
.گزینش نام متغیر
    .هنگامیکه ۱۲ { کاری را انجام بده }
    .هنگامیکه «مقدار یک رشته»
        کار دیگری را انجام بده()
    .هنگامیکه .در [۲..۵]
        }
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
کار سوم را انجام بده()
                    {
.هنگامیکه .در [۲..۵::۱/۰]
کار چهارم را انجام بده()
.وگرنه
کار پیشفرض را انجام بده()
```

۱۵.۵. ارجاع‌ها

در این زبان ارجاع‌ها به کمک & انجام می‌شود. به جای کپی شدن مقدار متغیر فقط نام دیگری برای متغیر اصلی در نظر گرفته می‌شود. مهم‌ترین کاربرد ارجاع‌ها در فراخوانی تابع‌ها است. در فراخوانی تابع‌ها همواره مقدار آرگومان‌ها کپی می‌شود ولی متغیرهای ارجاعی کپی نمی‌شوند.

```
متغیر ارجاعی = & متغیر معمولی
```

انتساب یک متغیر ارجاعی به یک متغیر دیگر یک ارجاع دیگر فراهم می‌کند. متغیرهای ارجاعی در این زبان مانند متغیرهای از نوع کلاس‌ها در زبان‌های java و C Sharp است و همان ویژگی‌ها را دارد.

```
متغیر ارجاعی ۲ = متغیر ارجاعی
```

برای اینکه کپی واقعی انجام شود باید عملگر یگانی * به کار رود.

```
متغیر۳ = * متغیر ارجاعی
```

۱۶.۵. آرایه‌ها

در این زبان آرایه‌ها از یک آغاز می‌شوند تا کار با آن‌ها ساده‌تر گردد و دشواری‌های آغاز با صفر را نداشته باشند. مانند زبان matlab می‌توان آرایه‌ها را با هم جمع، تفریق و ضرب کرد به شرطی که از نظر بعد همخوانی داشته باشند. یک عدد می‌تواند در یک ارایه (یک یا چند بعدی) ضرب شود یا با آن جمع شود یا از آن کم شود.

میان آرایه‌ها و نگاشت‌ها در پیاده‌سازی داخلی تفاوت وجود دارد ولی به صورت ضمنی هر جا نیاز باشد تبدیل انجام می‌شود یعنی هرجا شدنی بود نگاشت به آرایه تبدیل می‌شود و به صورت آرایه با آن رفتار می‌شود و برعکس آن همیشه شدنی است یعنی همواره یک آرایه را می‌توان به نگاشت تبدیل کرد. نمایش داخلی از چیزی که برنامه نویس می‌بیند پنهان است.

کروشه‌ی باز باید به نام آرایه بچسبد و نباید از آن سوا باشد.

مانند زبان php آرایه‌ها می‌توانند به سادگی و بدون نیاز به تعریف به کار برده شوند.

```
آرایه ۱[]=۱۲
آرایه ۱[]=۱۴
```

```
array1[]=12
```

```
array1[]=14
```

دو خط بالا مانند دو خط زیر عمل می‌کند.

```
آرایه ۱[۱]=۱۲
آرایه ۱[۲]=۱۴
```

```
array1[1]=12
array1[2]=14
```

می‌توان یک‌جا چندین مقدار درون آرایه گذاشت.

```
آرایه ۱=[۱۴،۱۲]
```

```
array1=[12 , 14]
```

می‌توان آرایه را به شکل زیر خالی کرد.

```
آرایه ۱=[]
```

```
array1=[]
```

اگر به صورت زیر مقدار دهی انجام شود فقط یک عنصر درون آرایه گذاشته می‌شود که خود یک آرایه است.

```
آرایه ۱[]=[۱۴،۱۲]
```

```
array1[]=[12,14]
```

در این حالت آرایه ۱ فقط یک خانه با نمایه‌ی (index) یک دارد که درون آن یک آرایه با دو عنصر گذاشته شده است.

نمایه‌های آرایه می‌توانند هر نوعی باشند و درواقع به صورت نگاشت باشند. همچنین کروشه‌های درونی مفهوم آرایه‌های داخلی را دارند و با آرایه‌های چند بُعدی تفاوت دارند.

```
B1 = [ [2,4],[3,5],[6,15],[12.3,6] ]
```

آرایه‌ی بالا یک آرایه با ۴ عنصر است که هر عنصر آن آرایه‌ای با دو عنصر را در بردارد.

```
B2[“ali”] = “reza”
B3[2,4]=5
```

آرایه‌هایی که مانند زبان C باشند به صورت زیر تعریف می‌شوند.

```
آرایه ۱[۱..۳::۱]=[۰،۰،۰]
آرایه ۱[۳..] = [۰]
آغاز = ۱
پایان = ۳
گام = ۱
یک عدد = ۰
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
آرایه ۱[آغاز..پایان::گام] = [یک یا چند عدد یا بازه‌ی عدد]
```

```
A1[1..3:1] = [0,0,0] //  A1[1]=0.....A1[3]=0
A1[..3]=[0]   //  A1[1]=0  , A1[2]=0  , A1[3]=0
start=1;
end =3
step=1
val =0
A1[start..end:step]=[val]
```

مانند زبان C نیز می‌توان مقدارهایی را برای آن در نظر گرفت و آرایه را تعریف کرد.

دو دستور زیر یک کار را انجام می‌دهند. خانه‌ی ۱ و ۱ از آرایه‌ی A1 برابر ۳ گذاشته شده است. و یک آرایه‌ی دو بُعدی تعریف شده است.

```
A1[1,1]=3
A1[,]=3
```

آرایه‌های چند بعدی در این زبان مانند دستور بالا به کار برده می‌شوند و به این ترتیب تعداد کمتری از علامت برای دسترسی به عنصرهای آن نیاز است. همچنین نیازی نیست که نمایه‌های آرایه‌های چند بعدی عددی باشند بلکه می‌توانند هر نوعی باشند.

```
A1["ali","reze"] = "hamid"
```

```
فهرست دوستان[«علی» ؛ «رضایی»] = «حمیدی»
```

همچنین یک آرایه یا فهرست از اشیاء (عدد، نویسه، رشته، ....) را می‌توان به عنوان نمایه‌های یک آرایه معرفی کرد.

```
L1 = ["Ali", "Reza","Parsa", "Siavash"]
A1[L1]=0
A2[["a","b",0,5,-12,"Hamid"]]=0
// یک آرایه با نمایه‌های داده شده درون کروشه باز و بسته است که همگی مقدار اولیه‌ی صفر را می‌گیرند.
```

```
ف۱=[«علی»؛ «رضا» ؛ «پارسا» ؛ «سیاوش»]
آ۱[ف۱]= ۰
آ۲[[«ا» ؛ «ب» ؛ ۰ ؛ ۵ ؛ -۱۲ ؛ «حمید»] ] = ۰
```

می‌توان آرایه‌های چند بُعدی را نیز در آغاز تعریف کرد و مقدار دهی اولیه نمود. دو دستور زیر هر دو یک کار را انجام می‌دهند.

```
آرایه ۱[۱::۳..۱ ؛ ۱::۴..۱]=[[۰؛۰؛۰]؛[۱؛۱؛۲؛۲؛۲؛۲]]
آرایه ۱[..۳؛..۴] = [[۰]؛[۱؛۱؛۲]]
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

۱۷.۵. تابع‌های آماده

در این بخش تعدادی تابع‌های آماده زبان نوشته شده است. هنگام فراخوانی هر تابعی باید پرانتز باز به نام تابع چسبیده باشد و از آن سوا نباشد.

۱.۱۷.۵. تابع‌های عمومی

Function name	نام تابع	Function name	نام تابع
print	بنویس	len	طول
input	بخوان	size	اندازه
str	به رشته	system	سیستم
tstr	از رشته		

۲.۱۷.۵. تابع‌های ریاضی

Function name	نام تابع	Function name	نام تابع
sin		asin	
cos		acos	
tg		atg	
ctg		actg	
sinh		asinh	
cosh		acosh	
tgh		atgh	
ctgh		actgh	
log		ln	
truncate		round	
floor		ceil	
abs	قدر مطلق	sqrt	جذر
pow	توان		

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

۳.۱۷.۵. تابع‌های آماری

نام تابع	Function name	نام تابع	Function name
میانگین	avg	کوچکترین	min
انحراف معیار	stdv	بزرگترین	max
تصادفی	random	میانه	mode

۴.۱۷.۵. تابع‌های عضو رشته

نام تابع	Function name	نام تابع	Function name
نمایه	index	زیر رشته	substr
شکستن	split	نمایه وارون	rindex

۵.۱۷.۵. تابع‌های عضو آرایه‌ها

نام تابع	Function name	نام تابع	Function name
حذف	del	پیوند	join
افزودن	insert	اتصال	append
مرتب کلید	sortkey	مرتب	sort
جستجوی وارون	rfind	جستجو	find
		جستجوی همه	afind

۱۸.۵. تابع

نوشتن تابع یکی از بخش‌های مهم هر زبان برنامه نویسی است. در این زبان نیز تابع را می‌توان به سادگی به کمک کلمه‌ی کلیدی تابع (function) نوشت و آرگومان‌هایی را برای آن فرستاد. آرگومان‌ها و مقدار برگشتی تابع کپی می‌شوند مگر برای آن‌هایی که ارجاع برایشان گذاشته شده باشد. کلمه‌ی کلیدی **برگرد** (**return**) برای برگرداندن مقدار برگشتی تابع به کار می‌رود.

```
.تابع افزایش یک (عدد ۱) {
    .برگرد عدد۱ + ۱
```

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
{
#
.تابع افزایش دو (عدد ۱)
    .برگرد عدد ۱ + ۲
#
.تابع نوشتن عددهای اول (تا عدد){
    .اگر تا عدد < ۲
        .برگرد
    عدد ۱ = ۳
    بنویس(۲)
.تا هنگامی‌که عدد ۱ <= تا عدد {
        عدد ۲ = ۳
    پرچم ۱ = .نادرست
.تا هنگامی که عدد ۲ <= جذر(عدد ۱){
    .اگر عدد ۱ % عدد ۲ == ۰{
        پرچم ۱ = .درست
        .بشکن
            {
        عدد ۲+=۲
            {
.اگر پرچم ۱ == .نادرست
    بنویس(عدد ۱)
    عدد ۱ +=  ۲
        {
{
```

۱۹.۵. تبدیل به رشته

تابع رشته همه چیز را می‌تواند به رشته تبدیل کند. اگر درون یک شیء تابع رشته باز تعریف شده باشد از آن کمک می‌گیرد و گرنه یکایک عنصرهای شیء را به رشته تبدیل کرده و با فاصله کنار هم می‌گذارد.

۲۰.۵. تبدیل از رشته

تابع «از رشته» یک رشته را می‌گیرد و به هر نوع دلخواهی تبدیل می‌کند. اگر تابع از رشته در یک شیء سربار گذاری شده باشد که همان تابع را به کار می‌برد و گرنه به طور خودکار می‌کوشد به کمک فاصله‌ها یکی یکی جزءها را به دست آورده و تبدیل کند.

۲۱.۵. قانون بلاک‌ها

در این زبان مانند زبان C می‌توان برای دستورهایی که چندین خط را در بردارند آکولاد باز و بسته {} را به کار برد و برای آن‌هایی که یک خط را در بردارند نیازی به گذاشتن آکولاد باز و بسته نیست فقط باید به خط بعد برود. در اینجا مفهوم خط از مفهوم دستور جداست زیرا در یک خط می‌توان چندین دستور را گذاشت که با ؛ (یا نویسه‌ی دیگر مانند نقطه«.» یا «\»). در آینده یکی از این‌ها برگزیده خواهد شد.) از هم جدا شده‌اند و در پایان خط نویسه یا نویسه‌های پایان خط گذاشته می‌شود. بنابراین برای چند دستوری که درون یک خط هستند و با ؛ از هم جدا شده‌اند نیازی نیست که آکولاد باز و بسته گذاشته شود.

حتی تابعی نیز که یک خطی است، می‌تواند بدون آکولاد باز و بسته باشد.

برخلاف زبان python در این زبان نیازی نیست که تو رفتگی رعایت شود و بلاک به کمک تو رفتگی مشخص شود، پیشنهاد می‌شود همواره تورفتگی برای خوانایی بیشتر رعایت شود. همچنین محدودیت آن یک دستور بدون بلاک نیست بلکه یک خط است. بنابراین می‌توان همه‌ی برنامه را در یک خط نیز نوشت، گرچه پیشنهاد نمی‌شود.

۲۲.۵. پرونده‌ها

کار کردن با پرونده‌ها در این زبان (ParsPL) بسیار بسیار ساده است. چند گونه پرونده وجود دارد که هر کدام جداگانه توضیح داده می‌شوند.

۱.۲۲.۵. پرونده‌های متنی ساده

این پرونده‌ها با کدگذاری یونی‌کد با طول متغیر (utf-8) هستند. به این ترتیب که با باز کردن پرونده به سادگی می‌توان خط‌های درون آن را تغییر داد یا اینکه خطی را همانند کار با آرایه از آن خواند.

```
پرونده ۱ = باز کن(«نام پرونده»)
پرونده ۱[۱]= «سلام»
پرونده ۱[۲]= «حال شما چطوره ؟»
پرونده ۱[۳] = «امیدوارم تندرست باشید.»
پرونده ۱[۴] = «خدانگهدار»
بنویس(پرونده ۱[۲])
@{ خروجی به صورت زیر خواهد بود
 حال شما چطوره ؟
@}
پرونده ۱.ببند()
```

به طور خودکار در پایان خط‌ها نویسه (یا نویسه‌های) پایان خط گذاشته می‌شود و نیازی نیست این نویسه‌ها گذاشته شود. هنگامی که تابع پایان می‌پذیرد یا اینکه برنامه پایان می‌پذیرد پرونده به طور خودکار بسته می‌شود.

اگر درون متغیری که در بردارنده‌ی پرونده است چیزی دیگری گذاشته شود نیز پرونده به طور خودکار بسته می‌شود.

در هر جایی از پرونده می‌توان نوشت و بنابراین محدود به طولی که اکنون دارد نیست. برای خط‌های خالی فقط نویسه خط بعد گذاشته می‌شود. ولی برای خواندن از پرونده اگر بخواهد به خطی دسترسی پیدا کندکه جزء آن نیست آن‌گاه خطا می‌دهد. این خطا می‌تواند به صورت برگردان مقدار «پوچ» باشد یا اینکه یک استثناء رخ دهد. هنوز درباره‌ی اینکه کدام گزینه پیش‌فرض باشد تصمیم نگرفته‌ام.

هنگام نوشتن به طور خودکار نویسه‌های پایان خط افزوده می‌شوند و هنگام خواندن نویسه‌های پایان خط به طور خودکار برداشته می‌شوند و برنامه نویس نگران آن‌ها نخواهد بود.

برای اینکه بتوان به نویسه‌ی ویژه‌ای از یک خط نیز دسترسی پیدا کرد کار بسیار ساده است

بنویس(پرونده ۱[۲،۱])

دستور بالا «ل» را چاپ می‌کند.

برای دسترسی به خط‌ها می‌تواند دستور .چرخه را نیز به کار برد که به این ترتیب به سادگی خط‌ها را برمی‌گرداند. البته نمی‌تواند خط‌ها را در این حالت تغییر داد.

.چرخه خط .در باز کن(«نام پرونده»)

بنویس(خط)

۲.۲۲.۵. پرونده‌های دودویی ساده

در این زبان پرونده‌های دودویی ساده حالت ویژه‌ای از پرونده‌های دودویی هستند که مانند یک آرایه می‌توان با آن‌ها رفتار کرد و هر گونه نوعی را می‌توان درون آن‌ها گذاشت یا از درون آن‌ها حذف کرد. درواقع همه‌ی کارهایی که نیاز است که پرونده مانند یک آرایه رفتار کند در پشت صحنه انجام می‌شود و الگوریتم‌های ذخیره و بازیابی در آن به کار برده می‌شود تا به سادگی کاربر بتواند با آن کار کند.

پرونده ۱=باز کن(«نام پرونده»،«د»)

پرونده ۱[۱]= ۱۲

پرونده ۱[۲]= «سلام حال شما چطوره»

پرونده ۱[۳] = متغیر از هر نوعی

.چرخه متغیری .در پرونده ۱

بنویس(متغیری)

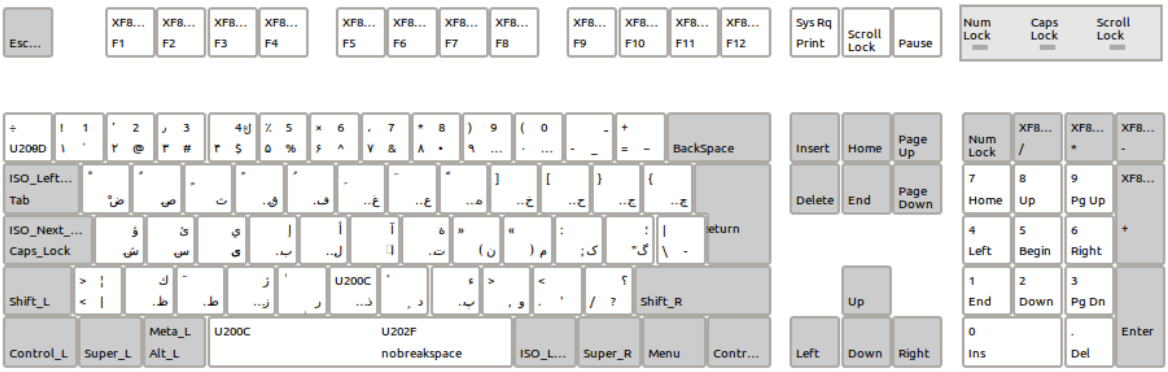
همانند کار کردن با یک آرایه می‌توان با این پرونده‌ها کار کـرد. در بخشـی از پرونـده نمایه‌هـا گذاشـته می‌شـود. همانند پایگاه‌های داده پرونده از همان آغاز نیز دربردارنده‌ی اندازه‌ای خواهد بود. اگر پرونده دربردانده‌ی

۲۳.۵. بررسی layout صفحه‌کلید فارسی در سیستم عامل‌های گوناگون

برای اینکه بتوان علامت‌هایی را در زبان به کار برد که به سادگی کاربر بتواند در سیستم عامل‌های گوناگون آن‌هـا را به کار ببرد layout صفحه‌کلید فارسی در سیستم عامل‌های گوناگون بررسی شـد. دقـت شـود کـه مهم‌تریـن نویسه‌ها آن‌هایی هستند که به سادگی بتوان آن‌ها را در صفحه‌کلید زد و همچنین در سیستم عامل‌های گونـاگون مکان یکسان و مناسبی داشته باشند و اگر میان انگلیسی و فارسی مشترک هم باشـند بسـیار بهـتر اسـت. بـرای نمونه نقطه (.) یکی از نویسه‌های مهم است. به همین دلیل چند کار برعهده‌ی آن گذاشته شد که در ادامه توضیح داده می‌شود.

Linux .۱.۲۳.۵

تصویرهای صفحه کلید فارسی در لینوکس که هماهنگ با استاندارد ایران است در زیر گذاشته شده است.

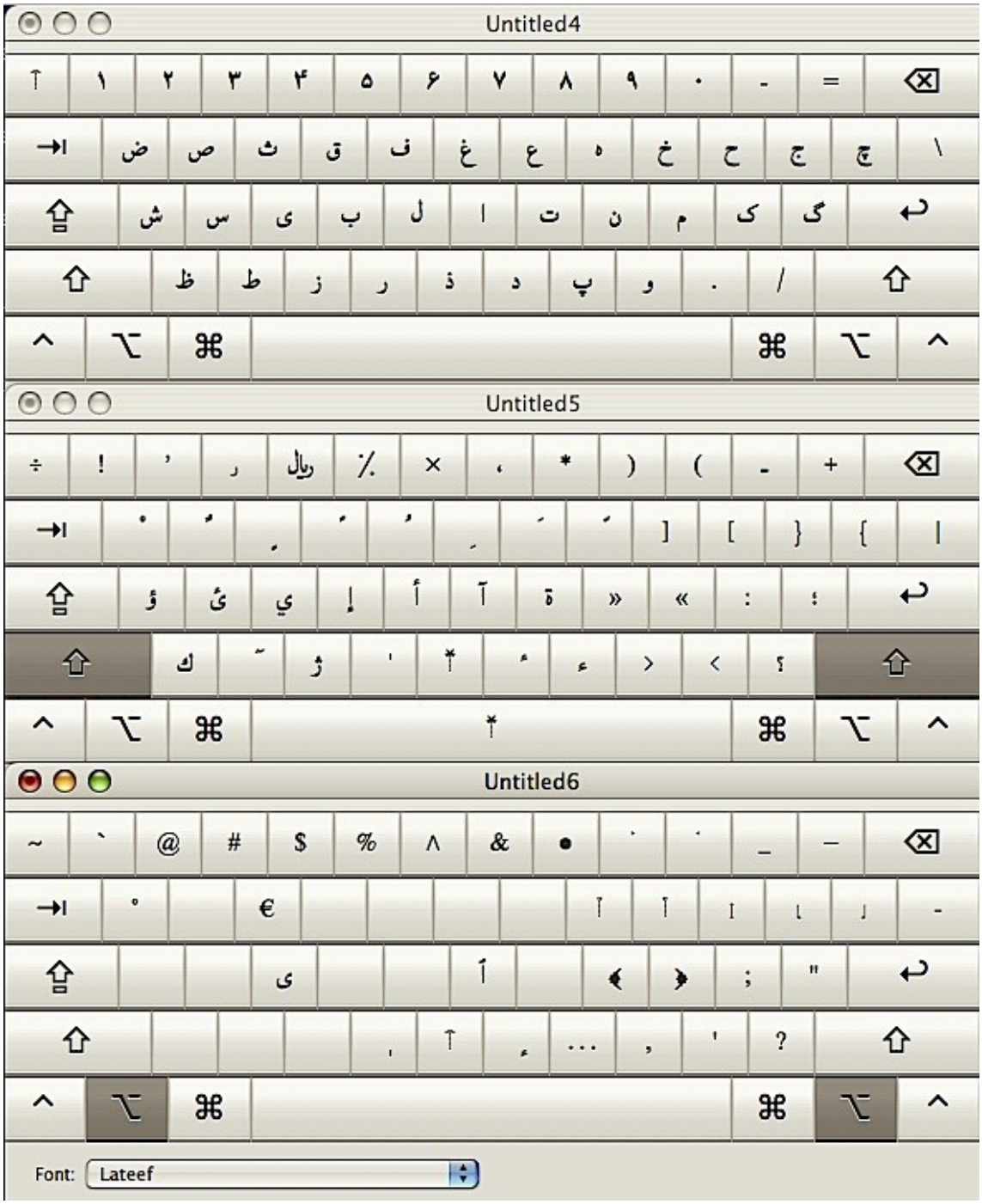


شکل ۱.۵: تصویر صفحه کلید فارسی در لینوکس

این تصویر از درون تنظیم‌هـای لینـوکس گرفتـه شـده اسـت. بـرای اینکـه بتـوان نویسـه‌هایی ماننـد $#@ %&^"؛'?, را در هنگام فارسی بودن صفحه‌کلید داشت (بدون تغییر صفحه کلید) بایـد کلیـد Right Alt را بـه همراه کلید مورد نظر گرفت. به این ترتیب به سادگی بسیاری از علامت‌هایی که نیاز است به دست می‌آید.

Macintosh .۲.۲۳.۵

تصویرهای صفحه کلید فارسی در لینوکس که هماهنگ با استاندارد ایران است در زیر گذاشته شده است.



شکل ۲.۵: تصویر صفحه کلید فارسی در سیستم عامل مکینتاش

این تصویر از سایت زیر گرفته شده است.

http://wiki.irmug.org/index.php/Persian-ISIRI-9147

همچنین در سایت زیر توضیح‌های سودمندی در این باره می‌توان یافت

http://www.hoomanb.com/Unicode/MacOSX.htm

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

Windows .۳.۲۳.۵

این سیستم عامل برخلاف دو سیستم عامل پیشین استاندارد صفحه‌ی کلید فارسی را رعایت نکرده است.

تصویرهای صفحه‌ی کلید فارسی در ویندوز xp در زیر گذاشته شده است.

شکل ۳.۵: تصویر صفحه کلید فارسی در ویندوز

شکل ۴.۵: تصویر صفحه کلید فارسی در ویندوز همراه با گرفتن shift

این شکل‌های از سایت زیر برداشته شده‌اند

http://msdn.microsoft.com/en-us/goglobal/bb964651.aspx

تصویر زیر که حالت ترکیبی را نشان می‌دهد.

شکل ۵.۵: تصویر ترکیبی صفحه کلید فارسی در ویندوز

این تصویر از سایت زیر گرفته شده است.

http://www.datacal.com/p-292-persian-farsi-keyboard-labels.aspx

همچنین در سایت‌های زیر صفحه کلیدهای گوناگون فارسی را در سیستم عامل‌های گوناگون بررسی کرده است.

http://tlt.its.psu.edu/suggestions/international/bylanguage/pashto.html
http://128.187.33.4/persian/persianword/kb.htm
http://128.187.33.4/persian/persianword/mac.html
http://www.farsiweb.ir

۴.۲۳.۵. نتیجه‌گیری

همان‌گونه که دیده می‌شود نویسه‌های . / \ = - نویسه‌هایی هستندکه در حالت فارسی بدون گرفتن shift به کار برده می‌شوند و تقریباً در هر سه سیستم عامل در مکان‌های یکسانی گذاشته شده‌اند. بنابراین این نویسه‌ها باید بیشترین کاربرد را داشته باشند. نویسه‌های / و - و = کاربرد خودشان را دارند و معنای آن‌ها تغییر نمی‌کند. نویسه‌ی سودمند دیگر که می‌تواند کاربرد بیشتری داشته باشد نویسه‌ی \ است که اکنون فقط درون یک رشته معنی‌دار است و بهتر است برای آن کاربرد بیشتری گذاشته شود. بهتر است علمگرها در زبان فارسی و زبان انگلیسی یکسان برگزیده شوند تا تغییرهای بسیار کمی روی زبان نیاز باشد و به سادگی بتوان به یک زبان دیگر نیز با آن برنامه نوشت و مجموعه عملگرها مستقل از زبان باشند.

یکی از مشکلات بزرگی که در راه تصمیم گیری درباره‌ی نشانه‌ها و عملگرهای زبان وجود داشته و وجود دارد این است که برخی از نشانه‌ها مانند «:» و «،» و «.» و «/» این مشکل را دارند که در هنگامی که با عددها ترکیب می‌شوند، ترتیب عددها به هم می‌ریزد و جابجا می‌شود مانند خط‌های زیر که درواقع عددها به ترتیب عکس وارد شده‌اند ولی نمایش متفاوتی یافته‌اند.

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

```
۱۲:۴۵:۵۶
۱،۲،۳
۱:۲:۳
```

این مشکل باعث می‌شود که برنامه نویس هنگام نوشتن برنامه دچار سردرگمی شود. در لینوکس در محیط gnome با gedit و ویندوز با notepad این مشکل بررسی شد و مانند هم بودند بنابراین باید چاره‌ی دیگری اندیشیده می‌شد. در زبان و محیط زبان «کلمات» ویرایشگر این مشکل‌ها را حل کرده است ولی اگر همان برنامه با ویرایشگر دیگری باز شود همین مشکلات را خواهد داشت. چون بر آن هستم که زبان وابسته به ویرایشگر نباشد و بتواند همه جا به درستی کار خود را انجام دهد بنابراین باید جایگزین‌های شایسته‌ای یافت می‌شد.

برخی از نویسه‌ها این مشکل را نداشتند مانند نقطه و نقطه «.» و نقطه ویرگول «؛» و «\» چون برای نقطه کاربردهای زیادی گذاشته شده بود نمی‌شد به جای کاما یا دو نقطه آن را به کار برد همچنین به کارگیری \ به جای کاما و دو نقطه چندان متداول نیست ولی شاید نقطه ویرگول گزینه‌ی بهتری باشد. همچنین نویسه‌های @ و ~ نیز نویسه‌های خوبی هستند که با عددها مشکلی ندارند و به خوبی از راست به چپ را رعایت می‌کنند. نوشتن @ در سیستم عامل‌های گوناگون ساده است ولی نوشتن ~ برای سیستم عامل ویندوز نیاز به تغییر زبان دارد. نویسه‌های # و $ ترتیب را به هم می‌ریزند. دونقطه : اگر پشت سرهم تکرار شود مشکلی نخواهد داشت و می‌توان آن را به کار برد بنابراین در آرایه‌ها به کار برده شد.

نقطه که شاید ساده‌ترین نویسه روی صفحه‌کلید برای فشرده شدن باشد باید کارهایی بیشتری در زبان انجام دهد به همین دلیل سه کاربرد گوناگون برای آن در نظر گرفته شده است.

۱. نقطه در آغاز یک کلمه‌ی کلیدی بیاید. پیش از آن باید یک فاصله یا خط جدید یا tab باشد. همچنین می‌تواند یک نقطه‌ی دیگر که نشان دهنده‌ی پایان دستور پیشین است؛ بیاید. همچنین }{() نیز همین وضعیت را دارند. باگذشت زمان با دقت بیشتری نویسه‌های بیشتری نیز در فهرست نویسه‌هایی که می‌توانند پیش از نقطه‌ی آغاز یک کلمه‌ی کلیدی بنشینند در نظر گرفته خواهد شد.

۲. نقطه در وسط در میان دو نویسه‌ای که نشان دهنده‌ی نام شناسه هستند به معنای دسترسی به متغیر یا تابع عضو یک شیء است نباید هیچ فاصله‌ای میان دو بخش گذاشته شود. دقت شود که فاصله‌ها و tab های آغاز و پایان شناسه‌ها حذف می‌شوند و در نظر گرفته نمی‌شوند.

۳. نقطه درون یک عدد اعشاری که دو طرف آن باید عدد باشد. حالت‌های زیر نادرست هستند.

```
12.
.12
12. 1
.
```

و باید به جای آن‌ها خط‌های پایین را گذاشت

```
12.0
0.12
```

```
12.1
0.0
```

برای علامت توان اعشاری همواره علامت ^ به کار برده می‌شود. این توان‌ها در همان تحلیل‌گر لغوی جایگزین عددشان می‌شوند و همان کاری را انجام می‌دهد که E یا e در زبانی مانند C انجام می‌دهد.

```
12.0^ 3.1
0.12 ^ 10
0.0 ^ 3
```

قانون‌هایی که برای حالت‌های بالا گذاشته شده است برای این است که هیچ ابهامی میان هر کدام از حالت‌ها نباشد و دستورهایی مانند دستورهای زیر معنای یکسانی بیابند.

```
i=j. a=b
// i=j.a=b  // error
i=j.a. k=12.
i=.j.   //Error, First dot indicate keyword, but .j is
not a keyword.
.if i==12 && j==13 {.if k==16 {   }.else{   }}
a.if.while //Correct. while is member of if and if
member of a.
```

۲۴.۵. برخی از پیشنهادهای کنار گذاشته شده

در این بخش برخی از پیشنهادهایی آورده شده است که در هنگام طراحی به ذهنم رسیده بود ولی به دلیل‌هایی کنار گذاشته شد.

۱ := و =: یا ترکیب هر عملگر دیگر با یک نماد که نشان دهنده‌ی کار آن از راست به چپ یا از چپ به راست باشد و مشکلی پیش نیاید.

۲ نقطه در پایان یک دستور به عنوان جدا کننده از دستور پس از آن. کاری که ; در بسیاری از زبان‌ها انجام می‌دهد. پس از این نقطه نیز باید } یا . یا space یا tab یا \n گذاشته شود. به دلیل ابهام‌هایی که کاربردهای گوناگون نقطه پیدا می‌کرد آن را برداشتم. اینکه نقطه را از آغاز کلمه‌های کلیدی بردارم مشکل‌ساز می‌شد و گزینش کلمه‌ها را برای برنامه نویس فارسی مشکل ساز می‌کرد ولی اکنون به سادگی می‌توان کلمه‌های کلیدی بیشتری را با آزادی بیشتری افزود بدون اینکه نگران اشتباه شدن آن‌ها با نام متغیرها شد. مانند بسیاری دیگر از زبان‌های برنامه نویسی ; جایگزین آن شد.

منابع

- Gerard O’Regan, A Brief History of Computing, Springer-Verlag London Limited 2008
- Paul E. Ceruzzi, "A History of Modern Computing", second edition 2003, The MIT Press, Cambridge, Massachusetts , London, England (This book was set in New Baskerville by Techset Composition Ltd., Salisbury, UK, and was printed and bound in the United States of America.)
- Abdulaziz Ghuloum, "An Incremental Approach to Compiler Construction", Department of Computer Science, Indiana University, Bloomington, IN 47408
- AHO, A. V., SETHI, R., AND ULLMAN, J. D. Compilers: Principles, Techniques, and Tools. 1986.
- APPEL, A. W. Modern Compiler Implementation in ML. Cambridge University Press, Cambridge, UK, 1998.
- ASHLEY, J. M., AND DYBVIG, R. K. An efficient implementation of multiple return values in scheme. In LISP and Functional Programming (1994), pp. 140–149.
- BURGER, R. G., WADDELL, O., AND DYBVIG, R. K. Register allocation using lazy saves, eager restores, and greedy shuffling. In SIGPLAN Conference on Programming Language Design and Implementation (1995), pp. 130–138.
- Niklaus Wirth, "Compiler Construction", ISBN 0-201-40353-6 Zürich, November 2005
- Dr. Dobb's, "Compiler Construction with ANTLR and Java", Journal March 1999
- [8] Anthony A. Aaby, "Compiler Construction using Flex and Bison", February 25, 2004

گزارش طرح پژوهشی «طراحی و پیاده سازی یک زبان برنامه نویسی کاملا فارسی»، احمد یوسفان ، شهریور ۱۳۹۰

- Erik Hilsdale, J. Michael Ashley, R. Kent Dybvig, Daniel P. Friedman, Bloomington, Indiana 47405
- H. Hamel, "Industrial Strength Compiler Construction with Equations ",S Software GmbH Guerickestr. 27.1000 Berlin10, Germany
- http://boostcon.boost.org/
- http://www.boostpro.com/
- http://boost-spirit.com/home/
- http://boost-spirit.com/home/people/
- http://www.boost.org/doc/libs/1_47_0/libs/spirit/doc/html/index.html
- http://www.boost.org/doc/libs/1_47_0/libs/spirit/example/qi/complex_number.cpp
- http://groups.google.com/group/comp.std.c++/topics
- http://en.wikibooks.org/wiki/Fortran/Fortran_examples
- http://stackoverflow.com/questions/5845219/fortran-sample-code-assemble
- http://www.personal.psu.edu/jhm/f90/progref.html
- http://en.wikipedia.org/wiki/Fortran
- http://en.wikipedia.org/wiki/Fortran_language_features
- http://fortranwiki.org/fortran/show/HomePage