آشنایی با نرم‌افزار رایانه

امین چاروسه
Charooseh@ce.aut.ac.ir

1. نرم افزار

واژه نرم افزار[1] در رایانه‌ها به برنامه‌های کنترل عملکرد سخت افزار اطلاق می شود. بدون نرم‌افزار سخت افزار تنها مجموعه‌ای از قطعات و تجهیزاتی است که قادر به انجام هیچ‌کاری نیست و با وجود نرم‌افزار است که سخت‌افزار جان می‌گیرد و می‌تواند کارهای مورد نیاز انسان را انجام دهد. برنامه‌ها در واقع دستورالعمل‌های قدم به قدم هستند که به رایانه می‌گویند چه عملی را در چه زمانی انجام دهد. نرم‌افزارها مجموعه‌ای از یک یا چند برنامه هستند که می‌توانند نحوه عملکرد رایانه‌ها را تغییر داده و باعث افزایش کارایی و قابلیت‌های آنها شوند.

2. انواع نرم‌افزار

نرم‌افزارها به طور کلی به دو دسته سیستمی و کاربردی تقسیم می شوند. نرم‌افزارهای سیستمی، برنامه‌هایی هستند که برای بهره‌برداری از سخت‌افزار، سایر نرم‌افزارها و مدیریت و پشتیبانی سیستم‌های رایانه‌ای و شبکه به کار گرفته می‌شوند. این نرم‌افزارها خود به دو دسته برنامه‌های توسعه سیستم و برنامه‌های مدیریت سیستم تقسیم می‌شوند. برنامه های مدیریت سیستم شامل سیستم‌های عامل و سیستمهای مدیریت پایگاه داده‌ها (DBMS[2]) و برنامه های توسعه سیستم شامل زبان‌های برنامه‌سازی، کامپایلرها و مترجم‌ها می‌باشند. در ادامه این فصل هر یک از این نمونه به اختصار مورد بررسی قرار می‌گیرند.

نرم‌افزارهای کاربردی که برنامه‌هایی جهت پردازش اطلاعات برای کاربران نهایی می‌باشند نیز به دو دسته برنامه‌های کاربردی تک منظوره و برنامه های کاربردی چند منظوره تقسیم می‌شوند. برنامه‌های تک منظوره در پاسخ به نیازهای خاص اشخاص یا سازمان‌ها تهیه می شوند. این برنامه‌ها ویژگی‌های خاصی برای حل مسائل مرتبط با کاری که برای آن طراحی شده اند، دارند. ولی برنامه های چند منظوره در کارهای که جنبه عمومی بیشتری دارد مورد استفاده قرار می‌گیرند. نرم‌افزارهای همه منظوره ضبط و نگهداری اطلاعات، محاسبات، گرافیک، ارتباطات و اساس و پایه کلیه کارهایی را که می‌توان با استفاده از کامپیوترهای شخصی انجام داد، تشکیل می‌دهند. نرم‌افزارهایی مانند: نرم‌افزارهای حسابداری، نرم‌افزارهای تخصصی و ... نمونه‌هایی از نرم‌افزارهای تک منظوره و مرورگرهای وب، صفحه گسترده‌ها، نرم‌افزارهای کار با پست الکترونیکی و واژه پردازها نمونه‌هایی از نرم‌افزارهای همه منظوره هستند.

3. سیستم عامل

اصلی ترین و مهمترین نرم افزار در یک کامپیوتر سیستم عامل است که مدیریت منابع سیستمی را برعهده دارد. سیستم عامل، سیستمی مجتمع از برنامه هاست که: فعالیتهای پردازنده را مدیریت می‌کند، ورودی، خروجی، حافظه و سایر منابع را کنترل می‌کند، امکانات لازم جهت اجرای برنامه های کاربردی را فراهم می‌کند، واسط بین کاربر و سخت افزار است و واسط بین کاربر و برنامه های کاربردی است.

در واقع سیستم عامل دارای فرمان هایی است که کاربر با اجرای هریک از آنها انجام عمل خاصی را از کامپیوتر می‌خواهد. سیستم عامل خود یک برنامه است که جهت اداره و کنترل کارها به حافظه RAM بار می‌شود و به این ترتیب قسمتی از حافظه را اشغال می کند و چون برای انجام وظایف خود می بایست در حافظه حضور دایمی داشته باشد به این دلیل می توان آن را مهمان همیشگی حافظه دانست.

اگر قرار بود برنامه کاربردی تنها به وسیله مجموعه ای از دستورالعمل های ماشین و با مسئولیت کامل سخت افزار ایجاد و کنترل شود، کار بسیار پیچیده و طاقت فرسا می‌شد. برای تسهیل کار مجموعه ای از برنامه های سیستمی تهیه شده است به بعضی از اینها برنامه سودمند می گویند برنامه

---

[1] Software
[2] Data Base Management Systems

های سودمند توابعی هستند که به دفعات مورد استفاده قرار می گیرند و به ایجاد برنامه، مدیریت پرونده ها و کنترل دستگاههای ورودی خروجی کمک می کنند برنامه ساز از این امکانات برای ایجاد برنامه کاربردی استفاده می کند و این کاربردها در هنگام اجرا این سخت افزار را از دید برنامه‌ساز پنهان کرده و رابط مناسبی را برای استفاده او از سیستم فراهم می کند. لذا سیستم عامل به صورت یک میانجی برای تسهیل دسترسی برنامه ساز و برنامه های کاربردی از امکانات و خدمات عمل می کند.

سیستم‌های عامل نیز به دو دسته تقسیم می‌شوند: سیستم‌های عامل تک وظیفه‌ای[1] در هر لحظه تنها یک برنامه را اجرا کرده و یا یک عمل انجام می‌دهند. در سیستم‌های عامل چند وظیفه‌ای[2] در هر لحظه بیش از یک برنامه را می‌توان اجرا نمود.

4. پایگاه های داده

آگاهی نسبت به ارزش استراتژیک اطلاعات سبب شد که داده‌ها و مدیریت آنها از اهمیت بیشتری برخوردار شوند. یک پایگاه داده، مجموعه‌ای از داده‌ها، یک ساختار مشخص و برنامه‌هایی است که قادرند مدیریت و پردازش داده‌ها را انجام دهند. قدرت بانک‌های اطلاعاتی به دلیل قابلیت انجام سه عمل مرتب کردن، جستجو و تهیه گزارشات است. یک سیستم مدیریت پایگاه داده از سه قسمت اصلی تشکیل شده است:

- یک زیر سیستم ذخیره‌سازی که برای ذخیره و بازیابی داده‌ها است.
- یک زیر سیستم مدل‌سازی و پردازش که روش‌هایی را برای سازمان‌دهی داده‌ها، افزایش، حذف، نگهداشت و به روزرسانی آنها ارائه می‌دهد.
- رابطی بین سیستم و کاربران.

5. مفسر و مترجم

برنامه‌ای که توسط برنامه‌نویس نوشته شده است را نمی‌توان مستقیماً توسط رایانه اجرا نمود. تمام زبان‌های برنامه‌سازی سطح بالا جهت ترجمه به زبان قابل فهم برای ماشین، نیاز به یک مترجم[3] یا مفسر[4] دارند. یک مفسر هر یک از دستورات برنامه را به یک یا چند دستورالعمل از زبان ماشین که فوراً قبال اجرا هستند، تبدیل می‌کند. این کار تا پایان برنامه برای هر یک از دستورات انجام می‌شود. اما یک مترجم اتدا زمانی را صرف بررسی کل برنامه کرده و سپس کلیه دستورات را به زبان ماشین تبدیل می‌کند. بدین ترتیب کل برنامه به یک برنامه اجرایی تبدیل شده و درحال اجرا نیز، دیگر نیازی به ترجمه خط به به خط آن نیست.

6. زبان‌های برنامه‌سازی

دستورالعمل‌هایی که برای رایانه نوشته می‌شود الگوریتم و به تشریح یک الگوریتم برای رایانه، برنامه رایانه‌ای می‌گویند. زبان برنامه‌سازی زبانی است که برنامه‌های رایانه‌ای با استفاده از آن نوشته می‌شوند. به طور کلی زبان‌های برنامه‌سازی بر اساس ساختار و نحوه ارتباط با کاربر و ماشین به پنج گروه و یا نسل تقسیم می‌شوند.

نسل اول: *زبان ماشین*- این زبان که از صفرها و یک‌ها تشکیل شده است نیاز به مترجم ندارد و مستقیماً برای ماشین قابل فهم است. چون برنامه‌نویس برای نوشتن برنامه به این زبان باید شناخت کافی نسبت به سخت‌افزار رایانه داشته باشد، نوشتن برنامه با این زبان مشکل بوده و بسیار وابسته به سخت افزار است.

نسل دوم: *زبانهای اسمبلی*[5]- مزیت اصلی زبان اسمبلی نسبت به زبان ماشین، استفاده از حروف و اختصارات به جای صفرها و یک‌ها است. هر دستور در زبان اسمبلی مستقیماً به یک دستور در زبان ماشین ترجمه می‌شود. زبان اسمبلی نیز وابسته به سخت‌افزار است. نوشتن برنامه به این زبان نیز مشکل ولی از زبان ماشین ساده‌تر است.

به زبان‌های سطح اول و دوم که به سخت‌افزار نزدیک‌تر هستند زبان‌های سطح پایین می‌گویند. زبان‌های نسل سوم به بعد زبان‌های سطح بالا نامیده می‌شوند. سطح بالا به این مفهوم است که این زبان‌ها به زبان انسان نزدیک‌تر هستند تا زبان ماشین.

نسل سوم: *زبانهای سطح بالا*- زبانهایی نزدیک به زبان انسان برای نوشتن برنامه های رایانهی مانند بیسیک، پاسکال و C هستند. این زبان‌ها همه منظوره بوده و برای حل مسائل عمومی به کار می‌روند. کاربر برای استفاده از این زبان‌ها نیاز به آموزش برنامه‌نویسی دارد.

نسل چهارم: *زبانهای مولد برنامه‌های کاربردی*- در این نسل کاربر برای به‌کار گیری نیاز به آموزش کمتری دارد. این زبان‌ها برای حل مسائل خاص طراحی شده‌اند.

نسل پنجم: *زبانهای طبیعی*- این زبان‌ها بسیار شبیه به زبان انسان‌ها هستند برای ایجاد ارتباط طبیعی‌تر با رایانه‌ها طراحی می‌شوند.

7. مهندسی نرم افزار

پیچیده‌تر شدن تولید نرم‌افزارها به مطرح شدن مهندسی نرم‌افزار منتهی شد. مهندسی نرم‌افزار یعنی به کارگیری اصول علمی و ریاضی جهت طراحی و تولید نرم‌افزار مبتنی بر تکنولوژی روز با کیفیت بالا.

---

1 Single Task
2 Multi Task
3 Compiler
4 Interpreter
5 Assembly

تولید کنندگان نرم‌افزار براین باورند که صرف نظر از روش‌ها و ابزارهای مورد استفاده، روش دوره حیات سیستم[1]، یک روند استاندارد برای تولید نرم افزار است. در این روند ابتدا نیازهای کاربر تعیین شده، طراحی صورت گرفته و پس از پیاده سازی و آزمایش نرم‌افزار مرحله نگهداشت آغاز می‌شود. مرحله نگهداشت برای رفع اشکالات احتمالی و برطرف کردن نیازهای جدید است. روند تولید نرم‌افزار از فازهای مختلف تشکیل شده است:

- تجزیه وتحلیل مسأله.
- طراحی.
- پیاده‌سازی.
- آزمایش.
- نگهداری نرم‌افزار.

در قدم اول مسأله باید به درستی تعریف و فهمیده شود و یک راه حل برای آن پیدا شود. تهیه فهرست نیازها با کمک کاربر به درک بهتر مسأله کمک می‌کند. این فهرست مشخص می‌کند که نرم‌افزار باید چه کاری انجام داده و چگونه سازماندهی شود.

سپس با استفاده از روش *طراحی* بالا به پایین طراحان مسائل پیچیده را به واحدهای کوچکتری تجزیه می‌کنند. پس از این مرحله طراح می‌تواند ارتباط میان این واحدها و جزئیات هر کدام را طراحی کند.

پس از تعریف مسأله و پیدا کردن راه‌حل، نوبت به *پیاده‌سازی* راه‌حل‌می‌رسد. این کار شامل نوشتن برنامه‌های کوچک، اشکال زدایی و مرتبط کردن آنها با یکدیگر است.

با اشکال‌زدایی تنها خطاهای برنامه‌ساز مشخص می‌شود. اما در طی مرحله *آزمایش* برنامه درستی عملکرد برنامه بر اساس معیارهای مورد نظر ارزیابی می‌شود. معمولاً برای این کار برنامه با استفاده از داده‌های مختلف و شبیه‌سازی شرایطی که برنامه نهایی تحت آن باید کار کند، آزمایش می‌شود، تا اطمینان حاصل شود که برنامه کارهای مورد نظر را انجام می‌دهد و کارهای ناخواسته را نیز انجام نمی‌دهد.

آخرین فاز تولید نرم‌افزار، *نگهداری* است. برای اشکال زدایی یا از بین بردن خطاها، افزایش کارایی، تطبیق با سخت‌افزارها و نرم‌افزارهای جدید و یا برطرف کردن نیازهای جدید لازم است که نرم‌افزار به طور منظم و پیوسته بازنگری شود. این عمل معمولاً پیچیده و پر هزینه است.

با پیشرفت حرفه مهندسی نرم‌افزار، مهندسی نرم‌افزار به کمک رایانه[2] مورد توجه قرار گرفت. به مجموعه‌ای از ابزارها که جهت خودکار کردن مراحل طراحی و تولید پروژه‌های نرم‌افزاری بزرگ و پیچیده به کار می‌رود، CASE Tools گفته می‌شود. این ابزارها، اغلب یک محیط کامل برای برای تمامی مراحل تولید نرم‌افزار فراهم می‌کنند.

8. برنامه نویسی ساخت‌یافته

مهمترین قسمت یک برنامه الگوریتم آن میباشد که در برنامه های کوچک با کمی سعی، یافتن آن اغلب موارد کار چندان مشکلی نیست. ولی با بزرگتر شدن برنامه ها میزان پیچیدگی برنامه رشدی بسیار سریع‌تر از اندازه برنامه پیدا می کند علاوه بر بزرگتر شدن برنامه‌ها مشکل دیگری هم وجود دارد. اکثر برنامه نویسان در قسمت خاصی نسبت به سایر قسمت ها مهارت بیشتری دارند و در مواردی هم ورزیدگی کمتری دارند. این مسئله در مورد برنامه های تک منظوره مشکل خاصی ایجاد نمی‌نماید. ولی امروزه کمتر برنامه موفقی را می توان یافت که به جنبه هایی مانند گرافیک، انیمیشن، ساده‌سازی برای استفاده کاربر، استفاده از بانکهای اطلاعاتی و ... توجهی نکرده باشد.

به همین دلیل برنامه نویسان سعی کردند برنامه های بزرگ خود را تقسیم کنند و هر قسمت توسط یک برنامه‌نویس نوشته شود و در آخر توسط شخصی تمام قسمتها به هم مرتبط شود. که در این روش هر برنامه نویس بدون دغدغه خاطر نسبت به سایر بخشها تنها به قسمتی که در آن دارای مهارت بیشتری است، می پردازد.

بی شک تقسیم برنامه به قسمتهایی مستقل و جدا از هم علاوه بر نیاز به یک کارشناس و دانش مرتبط به آن، به ابزارهایی نیز نیاز دارد تا به کمک آن اتصال برنامه ها به هم تا حد ممکن آسان و قابل اطمینان باشد. این عمل را می‌توان توسط نرم افزاری خاص انجام داد. ولی برنامه نویسان به این نتیجه رسیدند که اگر زبانهای برنامه نویسی را به گونه ای تغییر دهند که بتواند این تکنیک را خود انجام دهد؛ کاری بسیار معقولانه تر و عملی تر خواهد بود. به همین دلیل زبانهای برنامه نویسی ساخت یافته پدید آمدند.

برنامه‌سازی ساخت‌یافته روشی است که در آن برنامه به واحدهای کوچکی تقسیم شده و بدین ترتیب، درک مسأله و سازمان‌دهی آن آسان‌تر می‌شود. زبان‌های Pascal وC نمونه‌های خوبی از این زبان‌ها هستند. روند برنامه‌سازی ساخت‌یافته به گونه‌ای است که فعالیت‌ها و تصمیم‌گیری‌های فاز طراحی را طوری فرموله می‌کند که گروه‌های مختلفی از برنامه‌سازان و طراحان می‌توانند با هم بر روی پروژه‌های تولید نرم‌افزار کار کنند.

9. برنامه نویسی شئ گرا

1 System Life Time
2 Computer Aided Software Enginnering

علاوه بر برنامه‌سازی ساخت یافته، روش دیگری برای تولید نرم‌افزارهای پیچیده وجود دارد که برنامه نویسی شیءگرا[1] نامیده می‌شود. برنامه نویسی شیءگرا تکنیکی است که در آن طراح مسأله را به واحد[2]های کوچکتری به نام شیء تجزیه می‌کند. هر شیء داده‌ها و دستورالعمل‌های خاص خود را داشته و عمل خاصی را انجام می‌دهد. این روش سه ویژگی زیر را دارد:

- *استقلال داخلی*- قابلیت مدیریت موجودیت‌های نرم افزاری به نام شیء کهکه تنها به شکل های تعریف شده و قابل کنترلی با یکدیگر در ارتباط می‌باشند.
- *تشابه*- قابلیت تشریح موجودیت‌های نرم‌افزاری بر اساس تفاوت های آنها. به عبارت دیگر، یک شیء جدید می تواند ویژگی‌های یک شیء قدیمی را به ارث ببرد.
- *طبقه بندی*- قابلیت تعریف یک تعداد موجودیت تحت یک کلاس به خصوص که تماماً رفتار و ویژگی‌های مشابه دارند. مجموعه‌ای از کلاس‌ها که با یک محیط خاص در ارتباط باشند، کتابخانه[3] نامیده می‌شوند.

طرفداران برنامه‌سازی شیءگرا بر این باورند که این تکنیک سرعت بیشتری به روند تولید نرم‌افزار داده، هزینه تولید و نگهداری را کاهش داده و انعطاف پذیری زیادی برای تغییرات آتی در اختیار برنامه‌ساز قرار می‌دهد.

مقایسه برنامه‌سازی ساخت یافته با برنامه‌سازی شیءگرا

ساخت‌یافته	شیءگرا
تعریف داده‌ها و رویه‌ها تنها به صورت تصادفی ممکن است باهم مرتبط باشند.	تعریف داده‌ها و رویه‌ها به صورت یک شیء ترکیب می‌شوند.
داده‌ها قابل رؤیت می‌باشند.	داده‌ها معمولاً درون یک شیء پنهان می‌شوند.
داده‌ها و رویه‌هایی که رویه‌ها به روی آنها عمل می‌کنند، به طور صریح ترکیب می‌شوند.	برنامه‌ساز رویه‌ها را به‌طور کلی مشخص می‌کند و رویه‌ها توسط شیءها مورد استفاده قرار می‌گیرند.
نحوه کنترل به طور صریح توسط برنامه مشخص شده و در زمان ترجمه برنامه ثابت می‌شود.	نحوه کنترل در حین اجرا و توسط شیءها تعیین می‌شود.

[1] Object-Oriented
[2] Module
[3] Library