طراحی و پیاده سازی زبانهای برنامه نویسی

فصل اول :
اصول طراحی زبان ها

مدرس:
نیلوفر رنجبر

منبع :
http://smbidoki.ir/crsdetail.php?crsid=33

دانشگاه خلیج فارس
دانشکده فنی و مهندسی جم

چرا زبانهای برنامه سازی را مطالعه می کنیم؟

- برای بهبود توانایی خود در توسعه الگوریتمهای کارآمد
  - بازگشتی
- استفاده بهینه از زبان برنامه نویسی موجود
  - لیست ها، آرایه ها، رکورد ها، بازگشتی، کلاسهای اشیا و ...
- آشنایی با اصلاحات مفید ساختارهای برنامه نویسی زبانها
  - می توان کاستی های یک زبان را با استفاده از امکانات موجود پیاده سازی نمود.
- انتخاب بهترین زبان برنامه سازی
  - زبان مناسب برای پروژه خاص
- آموزش زبان جدید ساده می شود.
- طراحی زبان جدید ساده می شود.

تاریخچه مختصری از زبانهای برنامه سازی

- **زبانهای مبتنی بر اعداد (اواخر دهه ۱۹۳۰ تا اوایل دهه ۱۹۴۰)**
  - حل مسائل عددی به عنوان ماشین حساب های الکترونیکی
  - **اوایل ۱۹۵۰**
    - گریس هوپر زبان A-0
    - جان باکس کد گزاری سریع برای IBM 170
    - هدف: کامپایل عبارات محاسباتی ساده به زبان ماشین
  - **۱۹۵۵**، فرترن(جان باکس و گروهش)
  - ایجاد زبان الگوریتمی IAL(International Algorithmic Languages) به رهبری پیتر نائور
  - ALGOL 58 , ALGOL 60

زبانهای مبتنی بر اعداد ...

- مفهوم کلاسها در الگول و ارائه سیمولا۶۷(simula-67) در دهه ۱۹۶۰توسط نیگارد و دهل نروژی
- پاسکال توسط ویرث در دهه ۱۹۷۰(زبان علمی کامپیوتر)
- مفهوم کلاسها در C و ارائه C++ در دهه ۱۹۸۰توسط استراستراپ
- بیسیک برای محاسبات عددی عادی

تاریخچه مختصری از زبانهای برنامه سازی ...

- **زبانهای تجاری (۱۹۵۵)**
  - پردازش داده های تجاری، پس از محاسبات عددی به وجود آمدند.
  - ۱۹۵۵ هوپر، flowmatic
  - ۱۹۵۹ سازمان دفاع آمریکا، CBL
    - پس از توسعه منجر به طراحی زبان کوبول شد
    - استاندارد (1968)

تاریخچه مختصری از زبانهای برنامه سازی ...

- **زبانهای هوش مصنوعی (۱۹۵۰)**
  - IPL (سطح پایین)
  - LISP توسط جان مکارتی (زبان تابعی برای پردازش لیست ها)
    - جست و جو ، پردازش متن ، بازیهای کامپیوتری ، تفسیر ماشین خودکار
  - COMMIT توسط یاو
  - اسنوبال
  - پرولوگ

تاریخچه مختصری از زبانهای برنامه سازی ...

- **زبان های سیستمی**
  - تا مدتها استفاده از اسمبلی به دلیل نیاز به کارایی بالا
  - CPL
  - BCPL
  - Cدر پیاده سازی UNIX (1970)

تکامل معماری کامپیوتر و توسعه زبان ها

- دوران کامپیوترهای بزرگ
- دوران کامپیوتر شخصی
- دوران شبکه بندی

دوران کامپیوتر های بزرگ(دهه ۱۹۴۰ تا دهه ۱۹۷۰)

- **محیط دسته ای**
  - پردازش دسته ای: زبان مجموعه ای از فایل ها را به صورت ورودی دریافت می کرد، داده ها را پردازش می کرد و مجموعه ای از فایل های داده ای را به صورت خروجی تحویل می داد .(فرترن ، کوبول ، پاسکال)
- **محیط محاوره ای**
  - کاربر مستقیماً با برنامه در تعامل است، خروجی در نمایشگر، ورودی از کیبورد و ماوس
  - اشتراک زمانی به طوری که هر کاربر در برهه ای از زمان پردازنده کامپیوتر را در اختیار دارد.

تاثیر بر طراحی زبان

- زبانهای محیط دسته ای
  - فایل ها مهمترین ساختار ورودی خروجی
  - خطایی که اجرای برنامه را خاتمه می دهد، قابل قبول اما هزینه بر است پس پردازش خطا و استثنا مناسب است.
  - عدم وجود محدودیت زمانی بر روی سرعت اجرای برنامه
  - عدم وجود I/O محاوره ای
- زبانهای محیط محاوره ای
  - در صورت بروز خطا، خارج شدن از برنامه قابل قبول نیست. (اهمیت کم پردازش خطا)
  - برنامه محاوره ای باید از محدودیت زمانی برخوردار باشد.

دوران کامپیوتر های شخصی

- **کامپیوترهای شخصی دهه ۱۹۷۰**
  - مینی کامپیوترها و از دور خارج شدن کامپیوترهای بزرگ
  - Pc ها : تولید اپل ۲ به عنوان اولین نسل از PCهای تجاری در ۱۹۷۸

- **محیطهای سیستم تعبیه شده**
  - سیستمی کامپیوتری که برای کنترل بخشی از یک سیستم بزرگ به کار می رود.
  - خرابی سیستم هزینه گزافی دارد.
  - قابلیت اعتماد و صحت، صفات مهمی برای برنامه های تعبیه شده است.
  - C و C++

تاثیر بر طراحی زبان

- **کامپیوترهای شخصی**
  - واسط ویندوز(پنجره ها، لیست ها ، آیکن ها و ..)
  - نیاز به اشتراک زمانی کمتر و توسعه برنامه گرافیکی
  - نیاز به کارایی کمتر به علت استفاده فردی از کامپیوتر
- **سیستم های تعبیه شده**
- **کار با ورودی خروجی**
  - تعامل با I/O از طریق رویه هایی که به عنوان ویژگیهای دستگاه I/O محسوب می شود.
  - ارتباط با دستگاه های خاص، از طریق ویژگیهایی از زبان که **به ثبات های سخت افزاری ، محل های حافظه و یا پردازشگر وقفه** دسترسی دارند یا از **طریق زیربرنامه های نوشته شده به زبان اسمبلی یا یکی از زبان های سطح بالا**

سیستم های تعبیه شده (تاثیر بر طراحی زبان ...)

- **پردازش خطا**
  - بسیار پر اهمیت
  - فعالیت خودکار مناسب در جهت ترمیم سیستم و ادامه کار
  - کاربری برای برطرف کردن محاوره ای خطا وجود ندارد
- **زمان پاسخ دهی**
  - پاسخ در زمان محدود (بلادرنگ)
  - کارکرد سیستم کامپیوتری بزرگتر بسته به پاسخ سیستم تعبیه شده دارد
- **اغلب یک سیستم توزیعی است**
  - برنامه متشکل از مجموعه ای از کارهای همزمان و نامتناهی است

دوران شبکه بندی

- **محاسبات توزیعی(در اثنای دهه ۱۹۸۰)**
  - استفاده از کامپیوتر های مرکزی در شرکت ها برای پردازش داده های مشترک
  - شبکه های محلی و محاسبات کارگزار مشتری (مثال : سیستم رزرو هواپیما)
- **اینترنت(در اواسط دهه ۱۹۹۰)**
- **تاثیر بر زبانهای برنامه سازی**
  - توزیع اطلاعات کارگزاران بزرگ در سطح جهانی
  - زبانی برای تعامل بین کامپیوتر مشتری و کارگزار
  - امنیت صفحات
  - پردازش در سایت کاربر به علت سرعت محدود
    - خطوط ارتباطی
    - قدرت پردازش سرور در زمان دسترسی همزمان کاربران

صفات یک زبان خوب

صفات یک زبان خوب

- **وضوح، سادگی و یکپارچگی**
  - زبان باید مجموعه ای مفاهیم واضح، ساده و یکپارچه برای طراحی الگوریتم داشته باشد.
  - **جامعیت مفهومی:** مفاهیم مختلف و قوانین ترکیب آنها در یک زبان برنامه سازی
  - **خوانایی برنامه:** تفاوتهای نحوی باید منعکس کننده تفاوتهای معنایی باشد.
- **قابلیت تعامد:** امکان ترکیب ویژگیهای مختلف زبان و با معنا بودن ترکیب حاصل
  - مثال : ترکیب عبارت و ساختار شرطی
  - **مزیت :** یادگیری زبان ساده و نوشتن برنامه راحت
  - **معایب :** کامپایل بدون خطا در ترکیبهایی که منطق روشن و اجرای کارآمدی ندارند.
- **طبیعی بودن برای کاربردها**
  - زبانها باید ساختمان داده، عملگرها، دستورات کنترلی و نحو مناسب برای مسئله ای که باید حل شود را داشته باشند.
  - زبانی که برای کاربرد خاصی طبیعی است، ایجاد برنامه در این زمینه را ساده می کند.

صفات یک زبان خوب ...

- **پشتیبانی از انتزاع**
  - امکان طراحی انتزاع های مناسب توسط برنامه نویس با استفاده از ویژگیهای اولیه زبان
- **سهولت در بازرسی برنامه**
  - سادگی ساختارهای نحوی و معنایی زبان موجب سهولت در بازرسی برنامه
- **محیط برنامه نویسی**
  - وجود ویراستارهای خاص، امکانات نگهداری و اصلاح نسخه های متفاوت
- **قابلیت حمل برنامه**
  - عدم وابستگی به ماشین خاص
- **هزینه استفاده**
  - **هزینه اجرای برنامه** : در برنامه های بزرگی که دفعات زیادی اجرا می شوند اهمیت دارد.
  - **هزینه ترجمه برنامه:** در برنامه هایی که به تعداد زیاد ترجمه میشود ولی کمتر اجرا میشوند (زبان c در آموزش)
  - **هزینه نگهداری برنامه:** هزینه های ترمیم خطا بعد از اجرا ،تغییر نیازمندی های برنامه با توسعه و تغییر سخت افزار و سیستم عامل و . .

نحو (syntax) و معنای زبان

- نحو زبان برنامه سازی، ظاهر آن زبان است.
- قواعد نحوی مشخص می کنند که دستورات، اعلانها و سایر ساختارهای زبان چگونه نوشته می شوند.
- معنای زبان همان مفهومی است که به ساختارهای نحوی زبان داده می شود.

مدلهای زبان

- **زبانهای دستوری**(imperative) **یا رویه ای:** زبانهای مبتنی بر فرمان یا دستورگرا (این مدل از سخت افزار پیروی میکند که مجموعه ای از دستورات را به ترتیب اجرا میکند) مانند بیسیک
- **زبانهای تابعی**(applicative) : با استفاده از توابع قبلی، توابع جدید و پیچیده تر ساخته می شود

  مانند ام ال و لیسپ (بعضی وقتها c)

  $\mathbf{function_n(\ldots(function_2(function_1(data))\ \ldots)}$
- **زبانهای قانونمند**(rule-based): شرایطی را بررسی می کنند و درصورت برقرار بودن آنها فعالیتی را انجام می دهند.
  - مانند پرولوگ

  $\mathbf{enable\ condition_1 \longrightarrow action_1}$
- **برنامه نویسی شیئ گرا**(object-oriented): اشیای پیچیده به عنوان بسطی از اشیای ساده ساخته می شوند و خواصی را از اشیای ساده به ارث می برند.

- **زبانهای دستوری یا رویه ای (Imperative)**

برنامه یک سری از دستورات پشت سرهم است که از بالا به پایین اجرا می‌شوند.

```
Statement 1;
Statement 2;
```

مانند : C, C++, Fortran, Algol, Cobol, PL/I, Ada, Smalltalk, Pascal

- **زبانهای کاربردی یا تابعی (Functional)**

توابع در کنار هم برنامه را تشکیل می‌دهند و قابلیت فراخوانی تودرتو دارند.

$Function_n(... Function_1 (data)...)$

مانند : Schema, ML, Lisp

- **مدل مبتنی برقانون (Rule-Base)**

برنامه‌ها به صورت مجموعه ای از قوانین پایه ای تجزیه مساله ، می‌باشند.

ساختمان این زبان مشابه ساختار if است که در صورت رخداد شرایطی ،قانونی اجرا می‌شود.

Enabling condition1 $\longrightarrow$ action1 = Action1 if enabling condition1

مانند : Prolog

- **زبانهای شیء گرا (Object Oriented)**

برنامه از تعدادی ماژول تشکیل شده است که هر ماژول مشابه یک برنامه در زبان C می‌باشد. مبحث اصلی در این نوع زبان پشتیبانی از کلاس است.

مانند : C++, Java, Visual

استاندارد سازی زبان

آیا این دستور در زبان C معتبر است و نتیجه چه خواهد بود؟

```
int i=(1&&2)+3
```

روش پی بردن به معنای دستورات:

- به مستندات زبان مراجعه شود.
- برنامه را در کامپیوتر تایپ و اجرا کرد.
- به استاندارد زبان مراجعه شود.

استانداردهای زبان دو دسته اند:

- **استاندارد خصوصی:** تعاریفی که توسط شرکت سازنده یا مالک زبان ارائه می شوند. (برای زبانهایی که کاربرد گسترده دارند مناسب نیست)
- **استاندارد عمومی:** تعاریفی که توسط سازمانهای مختلف به توافق رسیده اند و روشیست که بین پیاده سازیهای مختلف زبان، یکنواختی به وجود می آورد.

ANSI, IEEE, BSI, ISO,...

مسائل مهم در استفاده ی موثر از استاندارد

- **زمان شناسی:** چه زمانی باید زبان استاندارد شود؟
- **اطاعت و پیروی**
  - کامپایلر پیرو، کامپایلریست که وقتی یک زبان استاندارد به آن داده میشود یک خروجی درست تولید نماید.
  - برنامه نویس باید مراقب ویژگیهای اضافی که در کامپایلر وجود دارد باشد.
- **کهنگی:** کی استاندارد کهنه می شود و چگونه باید آن را اصلاح کرد؟
  - استاندارد باید با گذشته خود سازگاری داشته باشد.
  - برخی امکانات و ساختارهای زبان در مرور زمان کهنه یا مستهلک می شوند که برنامه نویسان جدید بهتر است از این ویژگیهای زبان استفاده نکنند.

محیط های برنامه نویسی

- **محیط برنامه نویسی:** محیطیست که برنامه در آن ایجاد و تست می شود.
- **محیط عملیاتی:** محیطیست که انتظار می رود برنامه در آن اجرا شود.

❖**محیط برنامه نویسی در دو زمینه بر طراحی زبان تاثیر گذاشته است:**

- ویژگیهای مربوط به کامپایل کردن مجزای زیربرنامه ها و سایر بخشهای برنامه
- ویژگیهایی که کار تست و اشکالزدایی برنامه را انجام می دهند

کامپایل مجزا

- مطلوبست در مونتاژ برنامه های بزرگ، بخشهای مختلف به صورت مجزا ترجمه، اجرا و تست شوند و سپس با هم ادغام گردند.
- **کامپایلر باید این اطلاعات را داشته باشد:**
  - مشخصه ی تعداد ، ترتیب و نوع پارامترهای زیربرنامه
  - اعلان نوع داده سراسری
  - تعریف نوع داده محلی خود زیر برنامه

- **نحوه تهیه این اطلاعات**

1. اعلان مجدد در زیر برنامه ها (فرترن)
2. ترتیب خاصی از کامپایل (ادا و پاسکال)
3. وجود کتابخانه های مربوطه (c++,Java)

تست و اشکال زدایی

- **ویژگیهای ردیابی اجرا**
  - برچسب زدن دستورات و متغیر های خارجی برای ردیابی
- **نقاط کنترلی (break point)**
  - کنترل برنامه به کاربر برمی گردد
- **ادعا (assertion)**
  - یک عبارت منطقی جهت بررسی شرایطی خاص در نقطه مورد نظر از برنامه است که در حین اجرا، در صورتی که این شرایط برقرار نباشد برنامه متوقف می شود یا روتین پردازش استثنا فراخوانی می گردد.

assert (x>0 and A=0) or (X=0 and A>B+10)