بسمه تعالی

طراحی و پیاده سازی زبانهای برنامه سازی

علی چوداری خسروشاهی

Akhosroshahi@IAUT.ac.ir

دانشگاه آزاد اسلامی

مرجع

اصول طراحی و پیاده سازی زبانهای برنامه سازی

تالیف: ترنسدبلیو. پرات – مارون وای. زیلکوویتز

فصل اول

اصول طراحی زبانها

چرا زبانهای برنامه سازی را مطالعه می کنیم؟

- برای بهبود توانایی خود در توسعه الگوریتمهای کارآمد
- استفاده بهینه از زبان برنامه نویسی موجود
- می توانید با اصلاحات مفید ساختارهای برنامه نویسی آشنا شوید.
- انتخاب بهترین زبان برنامه سازی
- آموزش زبان جدید ساده می شود.
- طراحی زبان جدید ساده می شود.

نقش زبانهای برنامه سازی

زبان خوب چگونه است؟

صفات یک زبان خوب

- وضوح، سادگی و یکپارچگی
- قابلیت تعامد (Orthogonality)
- طبیعی بودن برای کاربردها
- پشتیبانی از انتزاع (Abstraction)

نقش زبانهای برنامه سازی(ادامه)

زبان خوب چگونه است؟(ادامه)

صفات یک زبان خوب(ادامه)

- سهولت در بازرسی برنامه
- محیط برنامه نویسی
- قابلیت حمل برنامه
- هزینه استفاده
  - هزینه اجرای برنامه
  - هزینه ترجمه برنامه
  - هزینه نگهداری برنامه

نقش زبانهای برنامه سازی(ادامه)

زبان خوب چگونه است؟(ادامه)

نحو و معنای زبان

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

نقش زبانهای برنامه سازی(ادامه)

مدلهای زبان

- زبانهای دستوری(imperative)
  - زبانهای مبتنی بر فرمان یا دستورگرا
  - مثال: C و Pascal
- زبانهای تابعی(applicative)
  - به جای مشاهده تغییر حالت عملکرد برنامه دنبال می شود.
  - مثال: LISP و Scheme
- زبانهای قانونمند(rule based)
  - شرایطی را بررسی می کنند و درصورت برقرار بودن آنها فعالیتی را انجام می دهند.
  - مثال: Prolog
- برنامه نویسی شی گرا(object oriented)
  - اشیای پیچیده به عنوان بسطی از اشیای ساده ساخته می شوندو خواصی را از اشیای ساده به ارث می برند.
  - مثال: Java و C++

نقش زبانهای برنامه سازی(ادامه)

استاندارد سازی زبان

روش:

- برای پی بردن به معنای دستورات به مستندات زبان مراجعه شود.
- برنامه را در کامپیوتر تایپ و اجرا کنید.
- به استاندارد زبان مراجعه شود.

استاندارد خصوصی

استاندارد عمومی

مسائل مهم استفاده موثر از استاندارد:

- زمان شناسی (timeliness)
- اطاعت و پیروی (conformance)
- کهنگی (obsolescence)

فصل دوم

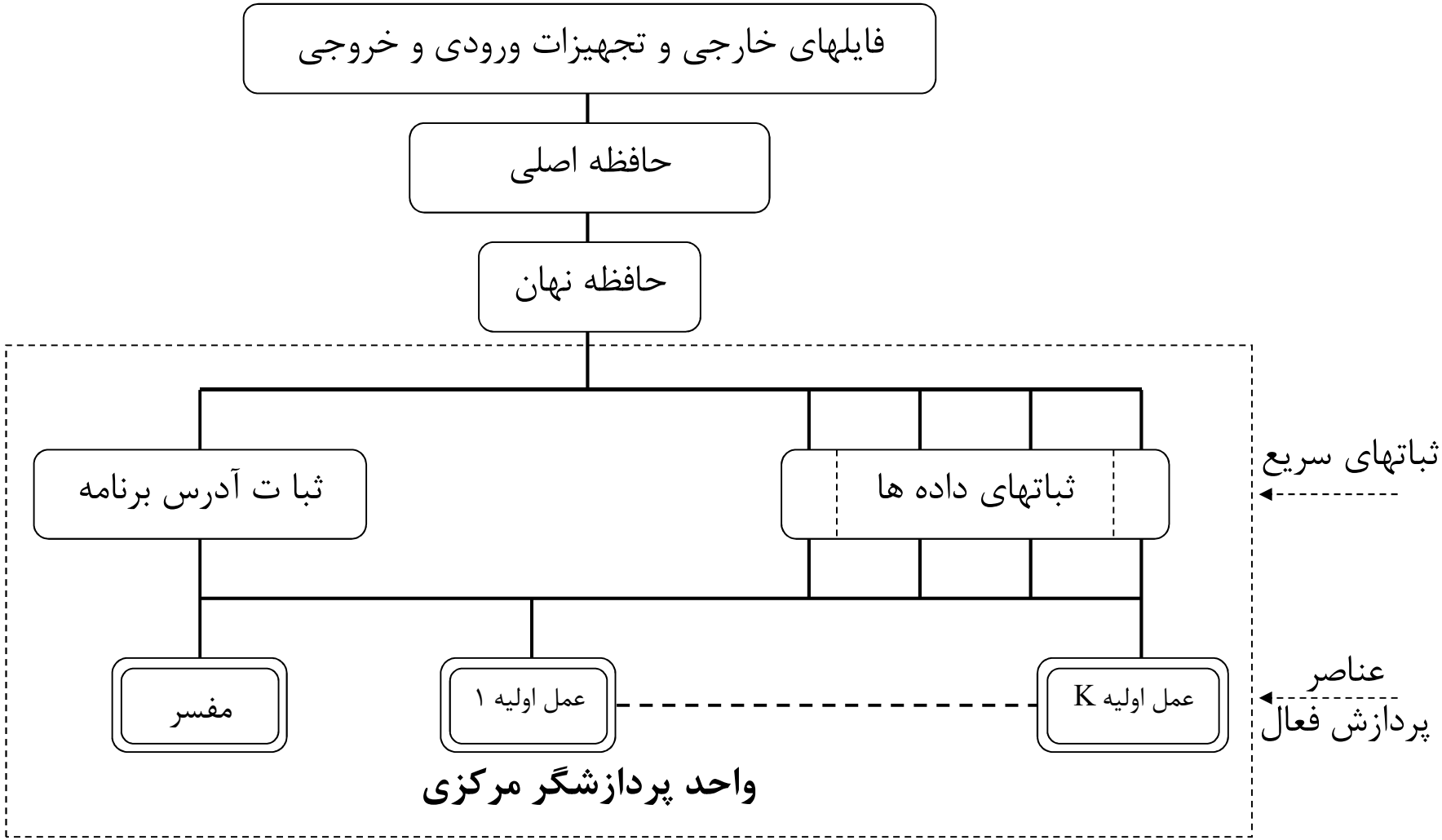
اثرات معماری ماشین

عملکرد کامپیوتر

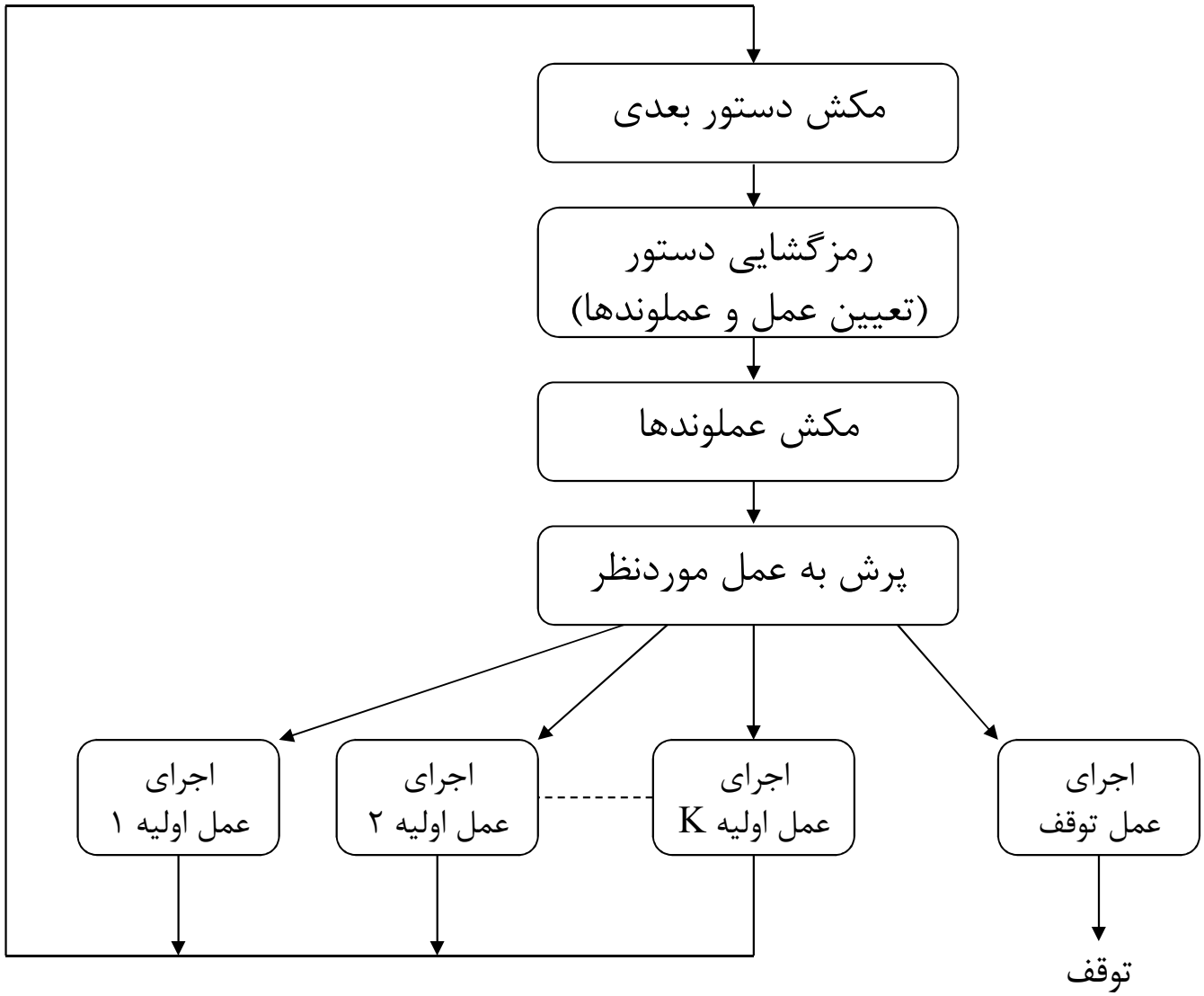
کامپیوتر مجموعه ای از الگوریتمها و ساختمان داده ها است که قابلیت ذخیره و اجرای برنامه ها را دارد.

- هر کامپیوتر از ٦ جزء تشکیل شده است:
  - داده ها (Data)
  - اعمال اولیه (Primitive operation)
  - کنترل ترتیب (Sequence control)
  - دستیابی به داده ها (Data Control)
  - مدیریت حافظه (Storage management)
  - محیط عملیاتی (Operating environment)

سازمان یک کامپیوتر معمولی



تفسیر و اجرای برنامه



عملکرد کامپیوتر (ادامه)

کامپیوترهای سخت افزار (hardware)

- کامپیوتر سخت افزار کامپیوتری است که کاملاً از اجزاء سخت افزاری و مدارات الکتریکی ساخته شده است. وضعیت ابتدایی برنامه مطابق با initial state کامپیوتر و نتیجه نهایی برنامه دقیقاً final state کامپیوتر است.

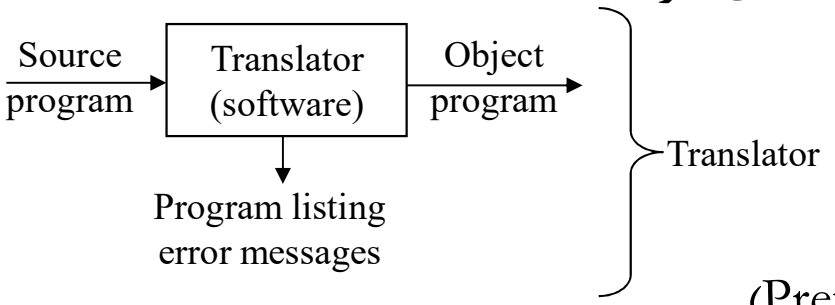
کامپیوترهای میان افزار (firmware)

- کامپیوتر میان افزار توسط ریز برنامه ای (microprogram) شبیه سازی می شود که بر روی کامپیوتر سخت افزار قابل ریزبرنامه نویسی اجرا می گردد. زبان ماشین آن مجموعه بسیار سطح پایین از ریز دستورات است که انتقال داده ها را بین حافظه اصلی و ثباتها، بین خود ثباتها و از ثباتها، از طریق پردازنده ها انجام می دهد.

عملکرد کامپیوتر (ادامه)

مفسرها و معماریهای مجازی

- ترجمه (کامپایل کردن) : مفسر می تواند طوری طراحی شود که برنامه ای به یک زبان سطح بالا را به برنامه ای در زبان ماشین ترجمه کند.
  - مفسر (Translator) هر پردازنده زبانی است که برنامه ای را به یک زبان منبع (که ممکن است سطح بالا یا پایین باشد) به عنوان ورودی گرفته به برنامه ای در زبان مقصد تبدیل می کند که از نظر کارایی با هم یکسان هستند.
  - انواع مفسرها را می توان بصورت زیر دسته بندی کرد:
    - اسمبلر (Assembler)
    - کامپایلر(Compiler)
    - بارکننده (loader)
    - پیوند دهنده (Linker)
    - پیش پردازنده یا پردازنده ماکرو (Preprocessor)



عملکرد کامپیوتر (ادامه)

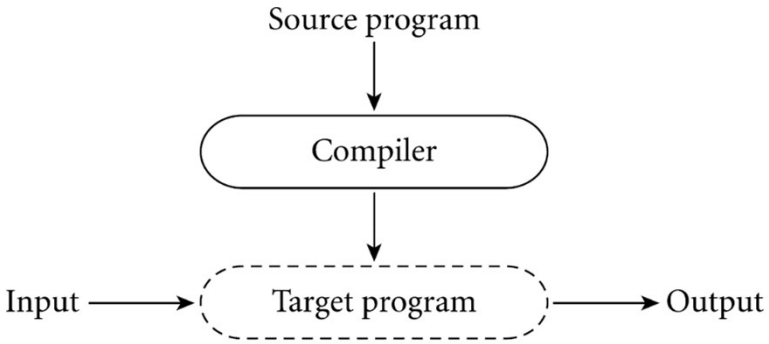
مفسرها و معماریهای مجازی (ادامه)

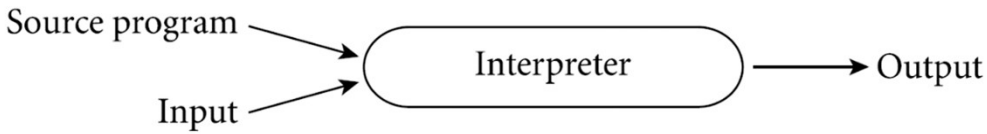
Software simulation (software interpreter)

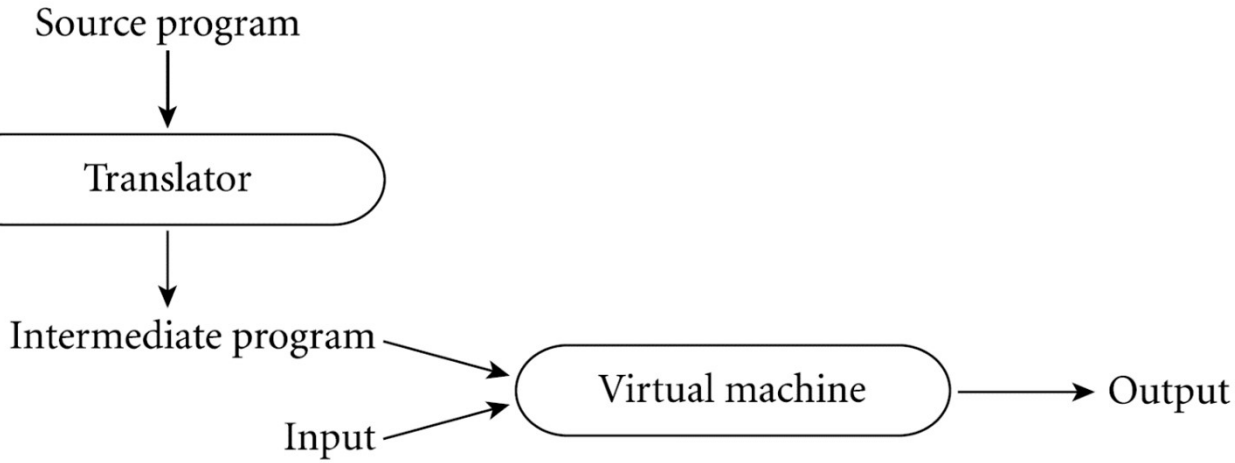
- شبیه سازی نرم افزاری (تفسیر نرم افزاری): به جای ترجمه برنامه های سطح بالا به برنامه های زبان ماشین معادل می توانیم از شبیه سازی استفاده کنیم که از طریق آن برنامه بر روی کامپیوتر میزبان اجرا می شود.

عملکرد کامپیوتر (ادامه)

مفسرها و معماریهای مجازی (ادامه)







کامپیوترهای مجازی و زمانهای انقیاد

روشهای ساخت کامپیوتر:

- از طریق سخت افزار (Through a hardware realization)
- از طریق نرم افزار (Through a firmware realization)
- از طریق ماشین مجازی (Through a software realization)
- از طریق ترکیبی (Through a combination above techniques)

کامپیوترهای مجازی و زمانهای انقیاد
(ادامه)

کامپیوترهای مجازی و پیاده سازی های زبان

- سه عامل منجر بر تفاوتهایی در بین پیاده سازیهای یک زبان می شود:
  - پیاده سازی های مختلف از کامپیوتر مجازی که به طور ضمنی در تعریف زبان وجود دارد، درک های متفاوتی اند.
  - تفاوتهایی در امکاناتی که توسط کامپیوتر میزبان ارائه می شود که زبان برنامه سازی باید بر روی آن پیاده سازی شود.
  - تفاوتها در انتخابهایی که توسط پیاده ساز صورت می گیرد تا عناصر کامپیوتر مجازی را با استفاده از امکاناتی که توسط کامپیوتر مربوط ارائه می شود پیاده سازی کند. علاوه بر این ساخت مترجم برای پشتیبانی از این انتخابهای نمایش کامپیوتر مجازی

کامپیوترهای مجازی و زمانهای انقیاد
(ادامه)
سلسله مراتب ماشینهای مجازی

کامپیوتر مجازی تعریف شده توسط برنامه نویس (پیاده سازی توسط زبان سطح بالا)
کامپیوتر مجازی زبان سطح بالا (پیاده سازی توسط برنامه ها و اجرا توسطOS)
کامپیوتر مجازی سیستم عامل (با برنامه های که بر روی کامپیوتر مجازی یا میان افزاری اجرا می شوند پیاده سازی می گردد)
کامپیوتر مجازی میان افزار (دستورات زبان ماشین پیاده سازی شده که با ریز دستورات توسط کامپیوتر واقعی اجرا می شود)
کامپیوتر سخت افزاری واقعی (توسط اجزاء فیزیکی پیاده سازی شده است)

کامپیوترهای مجازی و زمانهای انقیاد

انقیاد (Binding) و زمان انقیاد

- انقیاد یک عنصر برنامه به ویژگی یا صفت خاص، مثل انتخاب یک صفت از مجموعه ای از صفات است.
- زمانی که در حین فرموله کردن یا پردازش برنامه، این انتخاب صورت می گیرد، زمان انقیاد آن صفت برای آن عنصر نام دارد.

کامپیوترهای مجازی و زمانهای انقیاد

انقیاد (Binding) و زمان انقیاد (ادامه)

- زمان اجرا
  - در ورود به زیر برنامه یا بلوک
  - در نقطه خاصی از اجرای برنامه
- زمان ترجمه (زمان کامپایل)
  - انقیاد توسط برنامه نویس انتخاب می شود
  - انقیاد توسط مترجم انجام می شود
  - انقیادهایی که توسط بارکننده صورت می گیرد.
- زمان پیاده سازی زبان
- زمان تعریف زبان

کامپیوترهای مجازی و زمانهای انقیاد
(ادامه)

انقیاد (Binding) و زمان انقیاد (ادامه)

- در برنامه زیر که به زبان L نوشته می شود انقیادها و زمانهای انقیاد عناصر زیر بحث می شود:

```
X :=X+10
```

  - مجموعه ای از انواع ممکن برای متغیر X
  - نوع متغیر
  - مجموعه ای از مقادیر ممکن برای X
  - مقدار متغیر X
  - نمایش مقدار ثابت ۱۰
  - خواص عملگر +

کامپیوترهای مجازی و زمانهای انقیاد
(ادامه)

انقیاد (Binding) و زمان انقیاد (ادامه)

- زبانی مثل فرترن که در آن انقیاد در زمان ترجمه انجام می شود زبانهایی با **انقیاد زودرس** و زبانی با **انقیاد دیررس** مثل ام ال اغلب انقیادها را در زمان اجرا انجام می دهد.

فصل سوم

انواع داده اولیه

Elementary Data Type

E.D.T

خواص انواع و اشیاء

- هر برنامه صرفنظر از نوع زبان مجموعه ای از عملیات است که باید به ترتیب خاصی بر روی داده ها اجرا شوند.
- تفاوتهای بین زبانها ناشی از انواع داده ها، عملیات موجود و مکانیزم کنترل ترتیب اجرای عملیات بر روی داده ها است.

خواص انواع و اشیاء (ادامه)

اشیای داده ، متغیرها و ثوابت

- از اصطلاح Data object برای گروهبندی زمان اجرای یک یا چند قطعه از داده ها در کامپیوتر مجازی استفاده می کنیم.
  - Programmer defined
    - بعضی اشیا در حین اجرای برنامه توسط برنامه نویس تعریف شده اند.
    - مثل: متغیرها، مقادیر ثابت، آرایه ها و فایلها
  - System defined
    - بعضی از اشیای داده توسط سیستم تعریف می شوند.
    - اجزای تعریف شده توسط سیستم در حین اجرای برنامه در صورت نیاز به طور خودکار ایجاد می شوند.
    - مستقیما در اختیار برنامه نویس نیستند
    - مثل: پشته های زمان اجرا و بافرهای فایل

خواص انواع و اشیاء (ادامه)

اشیای داده ، متغیرها و ثوابت (ادامه)

- یک شئ داده، ظرفی (container) برای مقادیر داده است.
- یک شئ داده، توسط مجموعه ای از Attribute ها مشخص می شود
- یک شئ داده، می تواند Single number، Character، Pointer یا ... باشد.
- یک شئ داده، توسط Pattern ای از بیتها مشخص می شود.
- هر شئ داده ای دارای طول عمر (lifetime) است.
- اگر شئ داده حاوی مقداری باشد که همیشه به عنوان یک واحد (unit) دستکاری شود آن را شئ داده اولیه گویند.
- اگر شی داده مجموعه ای از سایر اشیای داده باشد ساختمان (struct) نامیده می شود.

خواص انواع و اشیاء (ادامه)

اشیای داده ، متغیرها و ثوابت (ادامه)

- انواع binding برای یک شئ داده:
  - نوع
  - محل (location)
  - مقدار
  - نام (name)
  - اجزاء (component)

خواص انواع و اشیاء (ادامه)

اشیای داده ، متغیرها و ثوابت (ادامه)

متغیرها و ثوابت

- شی داده ای که توسط برنامه نویس تعریف و نامگذاری می شود متغیر (variable) نام دارد.
- ثابت (constant) یک شی داده با نام است که مقداری به آن نسبت داده می شود.
  - یک ثابت لیترال (literal constant) ثابتی است که نامش همان نمایش مقدارش است
  - ثابت تعریف شده توسط برنامه نویس ثابتی است که نامش در تعریف شئ داده توسط برنامه نویس انتخاب می شود.

خواص انواع و اشیاء (ادامه)

انواع داده (Data Type)

- نوع داده طبقه ای از اشیای داده به همراه مجموعه ای از عملیات (manipulation) برای ایجاد و دستکاری آنها است.
- زبان برنامه سازی الزاماً با انواع داده هایی مثل دسته از آرایه ها ، مقادیر صحیح ، یا فایلها و عملیات مربوط به دستکاری آرایه ها ، مقادیر صحیح یا فایلها سروکار دارد.
- یک نوع داده ای در دو سطح مختلف بررسی می شود:
  - مشخصات (Specification)
  - پیاده سازی (Implementation)

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

- عناصر اصلی یک نوع داده درسطح Specification :
  - صفات (Attributes)
  - مقادیر (Values)
  - عملیات (Operations)
- عناصر اصلی یک نوع داده ای جهت پیاده سازی:
  - نمایش حافظه ای
  - روش دستکاری

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

- تقسیم بندی عملیات
  - Primitive operation
    - از دید برنامه نویس به دو دسته تقسیم می شود
      - عملیات دودویی (Binary operation)
      - عملیات یکانی (Unary operation)
  - Programmer defined operation
- معمولاً برای نمایش مشخصات عملیات از نشانه گذاریهای ریاضی استفاده می شود

op name : arg type × arg type × … arg type → result type

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

- چهار عامل موجب می شوند تا تعریف عملیات زبان برنامه سازی پیچیده شود:
  - عملیاتی که برای ورودیهای خاصی تعریف نشده اند.
  - آرگومانهای ضمنی (implicit arguments)
  - اثرات جانبی (نتایج ضمنی)
  - خود اصلاحی (Self-modification)

اگر نوعی به عنوان بخشی از نوع بزرگتر باشد آن را **زیر نوع** (subtype) و نوع بزرگتر را **ابر نوع** (supertype) می گویند.

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

پیاده سازی انواع داده اولیه

- نمایش حافظه
  - حافظه مربوط به انواع داده اولیه تحت تاثیر کامپیوتری است که برنامه را اجرا می کند.
  - با صفات اشیای داده اولیه به طور مشابه برخورد می شود:
    - برای کارایی، بعضی از زبانها طوری طراحی شدند که صفات داده ها توسط کامپایلر تعیین شوند.
    - صفات شی داده ممکن است در زمان اجرا دریک توصیف گر (descriptor) و به عنوان بخشی از شی داده ذخیره شود.

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

پیاده سازی انواع داده اولیه (ادامه)

- پیاده سازی عملیات
  - هر عملیاتی که برای نوعی از اشیای داده تعریف شد ممکن است به یکی از سه روش زیر پیاده سازی شود:
    - به صورت عملیات سخت افزاری
    - به صورت زیر برنامه رویه یا تابع
    - به صورت دستوراتی در داخل برنامه نوشته شوند.

خواص انواع و اشیاء (ادامه)

اعلانها (declaration)

- دستوری از برنامه است که نام و نوع اشیای داده و همچنین طول عمر آن را در حین اجرای برنامه مورد نیاز هستند مشخص می کند.
- اعلان می تواند به یکی از دو صورت انجام شود
  - صریح (explicit)
  - ضمنی (implicit)
- اطلاعاتی که توسط دستورهای اعلان آورده می شود
  - Name
  - Type
  - Initial value
  - Attribute

خواص انواع و اشیاء (ادامه)

اعلانها (ادامه)

اعلان عملیات

اعلانها می توانند اطلاعاتی راجع به عملیات را برای مترجم زبان فراهم کنند. برای عملیات باید

- تعداد پارامترها
- ترتیب پارامترها
- نوع پارامترها
- نتایج

خواص انواع و اشیاء (ادامه)

اعلانها (ادامه)

- اهداف اعلان:
  - انتخاب نمایش حافظه (storage representation)
  - مدریت حافظه (storage management)
  - عملیات چندریختی – چند شکلی (generic operation)
  - کنترل نوع و تبدیل نوع (type checking and type conversion)

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع

- منظور از کنترل نوع این است که هر عملیاتی که در برنامه انجام می گیرد تعداد و نوع آرگومانهای آن درست باشد.
- دو نوع کنترل نوع داریم
  - Dynamic type checking (D.T.C)
    - کنترل نوع ممکن است در زمان اجرا صورت گیرد(کنترل نوع پویا)
  - Static type checking (S.T.C)
    - کنترل نوع ممکن است در زمان ترجمه صورت گیرد(کنترل نوع ایستا)

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

- مزایای کنترل نوع پویا:
  - انعطاف پذیری زیاد در برنامه.
  - اعلان برای نوع می تواند نیاز نباشد.
  - عمل تبدیل نوع می تواند در زمان اجرا انجام شود.

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

- معایب کنترل نوع پویا:
  - اشکال زدایی (Debug) برنامه و حذف تمام خطاهای نوع آرگومان مشکل است.
  - در کنترل نوع پویا لازم است اطلاعات مربوط به نوع در زمان اجرای برنامه نگهداری شوند (مصرف حافظه).
  - کنترل نوع پویا باید به صورت نرم افزاری پیاده سازی شود (کاهش سرعت).

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

- مزایای کنترل نوع ایستا:
  - سرعت اجرای برنامه معمولاً زیاد است.
  - استفاده از حافظه بهینه است.
  - هیچ اطلاعاتی برای ذخیره نوع در خود برنامه نگهداری نمی کنیم.
  - تمام مسیرهای اجرای برنامه از جهت نوع چک می شود.

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

- معایب کنترل نوع ایستا:
  - انعطاف پذیری کم
  - هنگام مجزا کامپایل کردن برنامه ممکن است مشکلاتی بوجود آید.
  - در برخی زبانها به دلیل ساختار زبان کنترل نوع ایستا امکانپذیر نیست. در این زبان ها ممکن است به دو گونه عمل شود:
    - کنترل نوع پویا
    - عملیات کنترل نشوند.

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

- تبدیل نوع و تبدیل نوع ضمنی
  - عدم تطابق نوع ممکن است به عنوان خطا اعلان شود و فعالیت مناسبی صورت گیرد.
  - ممکن است تبدیل نوع ضمنی صورت گیرد تا نوع آرگومان واقعی به نوع درستی تغییر کند.

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

- تبدیل نوع و تبدیل نوع ضمنی (ادامه)
  - اغلب زبانها تبدیل نوع را به دو صورت انجام می دهند:
    - به صورت مجموع ای از توابع پیش ساخته که توسط برنامه نویس فراخوانی می شود تا بر تبدیل نوع اثر بگذارند.
    - در مواردی که عدم تطابق نوع صورت گرفت تبدیل ضمنی به طور خودکار فراخوانی می شود.

خواص انواع و اشیاء (ادامه)

انتساب و مقدار دهی اولیه (Assignment and Initialization)

- انتساب عملیات اصلی برای تغییر binding یک مقدار به یک شی داده است.
- این تغییر، اثر جنبی عملیات است.
- این دستور به دو شکل زیر می تواند انجام شود:

Assignment (:=): Type1*Type2 → Void

Assignment(=): Type1*Type2 → Type3

خواص انواع و اشیاء (ادامه)

انتساب و مقدار دهی اولیه (ادامه)

- تعریف عملیات انتساب به صورت زیر :
  - مقدار چپ اولین عبارت عملوند را محاسبه کن.
  - مقدار راست دومین عبارت عملوند را محاسبه کن.
  - مقدار راست محاسبه شده را به شئ داده مقدار چپ محاسبه شده نسبت بده.
  - مقدار راست محاسبه شده را به عنوان نتیجه عملیات برگردان.

خواص انواع و اشیاء (ادامه)

انتساب و مقدار دهی اولیه (ادامه)

- مقدار دهی اولیه: یک شی داده است که ایجاد شده ولی هنوز مقداری به آن داده نشده است.
  - متغیر های فاقد مقدار اولیه عامل مهمی برای بروز خطا در برنامه نویسی هستند.
- دو مقدار دهی اولیه وجود دارد:
  - صریح
  - ضمنی

انواع داده اسکالر
Scalar Data Type

- اشیایی وجود دارند که برای اشیای داده خود فقط یک صفت دارند.
  - مانند شئ نوع صحیح
- داده مرکب را در نظر می گیریم که درآن یک شی می تواند چندین صفت داده باشد.
  - مانند رشته کاراکتری
- اشیای اسکالر از معماری سخت افزار کامپیوتر پیروی می کنند.

انواع داده اسکالر(ادامه)

انواع داده عددی

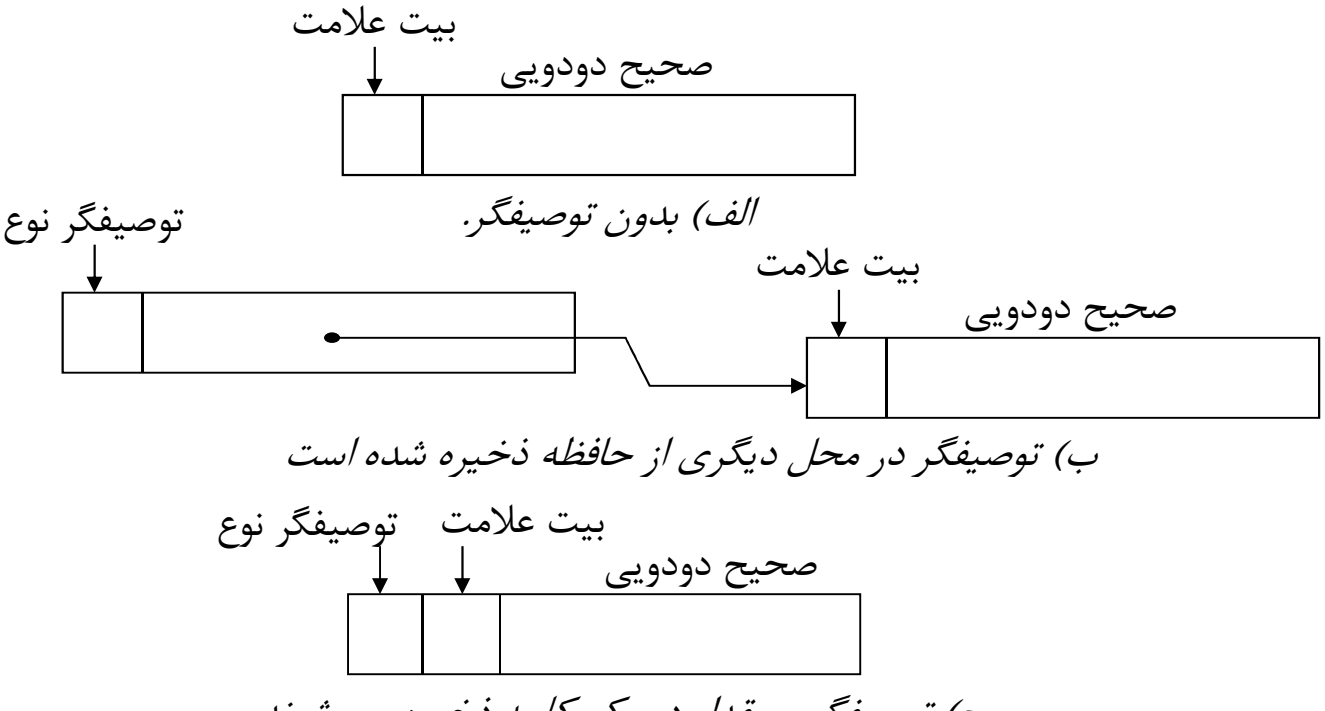
انواع صحیح :

- مشخصات : یک شی داده از نوع صحیح، معمولاً صفتی غیر از نوع ندارد.
- عملیات بر روی اشیای داده صحیح شامل موارد زیر است:
  - عملیات محاسباتی
  - عملیات رابطه ای
  - انتساب
  - عملیات بیتی

انواع داده اسکالر(ادامه)

انواع داده عددی (ادامه)

انواع صحیح : شکل زیر سه نمایش حافظه برای مقادیر صحیح را نشان می دهد.



الف) بدون توصیفگر.

ب) توصیفگر در محل دیگری از حافظه ذخیره شده است

ج) توصیفگر و مقدار در یک کلمه ذخیره می شوند.

انواع داده اسکالر(ادامه)

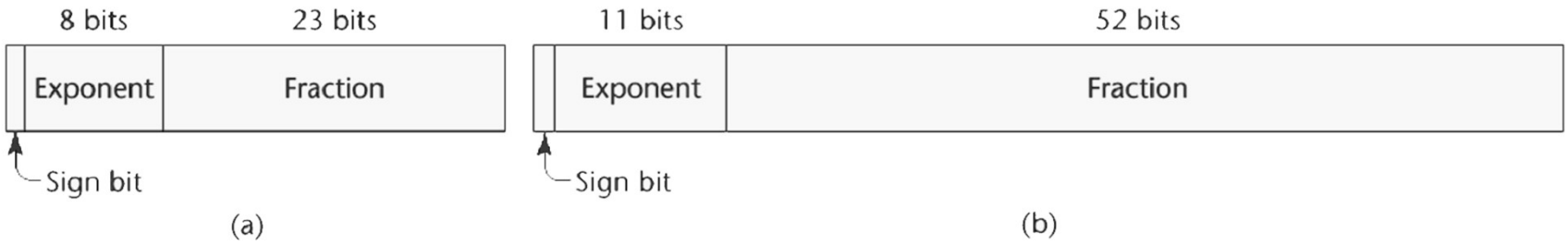
انواع داده عددی (ادامه)

- زیر بازه ها (subrange):
  - مشخصات : زیر بازه ای از نوع داده صحیح زیر نوعی از نوع داده صحیح است و شامل دنباله ای از مقادیر صحیح و بازه محدود است.
  - پیاده سازی: انواع زیربازه دو اثر مهم در پیاده سازی دارد:
    - نیاز به حافظه کمتر
    - کنترل نوع بهتر

انواع داده اسکالر(ادامه)

انواع داده عددی (ادامه)

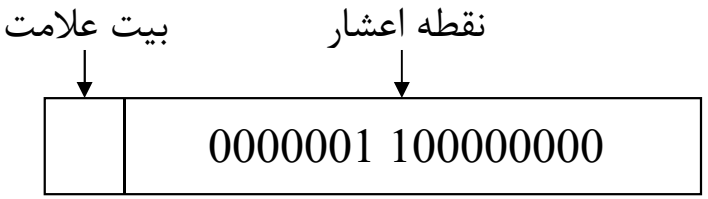
- اعداد حقیقی ممیز شناور (Floating Point Real Number)
  - مشخصات : معمولاً با صفت نوع داده مثل real در فرترن یا float در C مشخص می شود.
  - پیاده سازی: نمایشهای حافظه برای انواع آن معمولاً به سخت افزار بستگی دارد که در آن ممیز حافظه به دو بخش مانتیس (ارقام با ارزش عدد) و توان تقسیم می شود.
  - فرمت ممیز شناور IEEE 754



انواع داده اسکالر(ادامه)

انواع داده عددی (ادامه)

- اعداد حقیقی ممیز ثابت (Fixed point)
  - مشخصات : اغلب سخت افزارها شامل اشیائ داده صحیح و ممیز شناور هستند.
  - پیاده سازی: ممکن است مستقیماً توسط سخت افزار پشتیبانی شود یا به صورت نرم افزاری شبیه سازی گردد.



انواع داده اسکالر(ادامه)

انواع داده عددی (ادامه)

- سایر انواع داده عددی
  - *اعداد موهومی*: عدد موهومی متشکل از یک جفت از اعداد است که یکی از آنها بخش حقیقی و دیگری بخش موهومی را نشان می دهد.
  - *اعداد گویا*: عدد گویا خارج قسمت دو مقدار صحیح است.

انواع داده اسکالر(ادامه)

نوع شمارشی (Enumeration)

- مشخصات: لیست مرتبی از مقادیر مجزا است. برنامه نویس اسامی لیترالهایی را که باید برای مقادیر مورد استفاده قرار گیرند و همچنین ترتیب آنها را با استفاده از اعلانی مشخص می کند.
  - مثال: enum StudentClass {Fresh, Soph, Jonior, Senior};
- پیاده سازی: نمایش حافظه برای شی داده ای از نوع شمارشی ساده است.

انواع داده اسکالر(ادامه)

نوع بولی (Boolean)

- مشخصات: متشکل از اشیای داده ای است که یکی از دو مقدار TRUE یا FALSE را می پذیرد.
- پیاده سازی: نمایش حافظه برای شی داده بولی یک بیت از حافظه است .مقادیر true و false به دو روش در این واحد حافظه نمایش داده می شوند:
  - بیت خاصی برای این مقادیر استفاده می شود.
  - مقدار صفر در کل واحد حافظه نشاندهنده false و مقدار غیرصفر نشاندهنده true است.

انواع داده اسکالر(ادامه)

کاراکترها (Character)

- مشخصات : نوع داده کاراکتری اشیای داده را به وجود می آورد که مقدار آنها یک کاراکتر است.
- پیاده سازی: مقادیر داده های کاراکتری همیشه توسط سخت افزار و سیستم عامل پشتیبانی می شوند.

انواع داده مرکب

رشته های کاراکتری

مشخصات و نحو

- با رشته های کاراکتری حداقل به سه روش رفتار می شود:
  - طول ثابت
  - طول متغیر با حد بالا
  - طول نامحدود

انواع داده مرکب(ادامه)

رشته های کاراکتری (ادامه)

مشخصات و نحو (ادامه)

- عملیات گوناگونی بر روی رشته ها انجام پذیر است که بعضی از آنها عبارتند از:
  - الحاق
  - عملیات رابطه ای در رشته ها
  - انتخاب زیر رشته با استفاده از اندیس
  - فرمت بندی ورودی – خروجی
  - انتخاب زیر رشته با تطابق الگو (Pattern-matching)
  - رشته های پویا

.

انواع داده مرکب(ادامه)

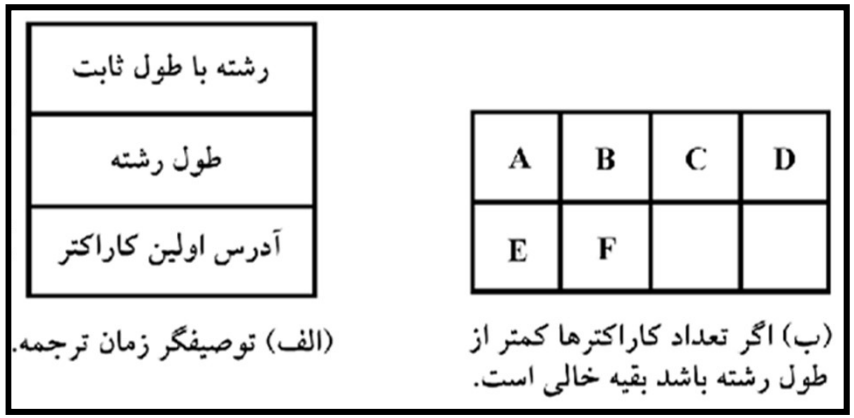
رشته های کاراکتری (ادامه)

پیاده سازی

- برای رشته ای با طول ثابت : نمایش حافظه همان شکلی است که برای بردار فشرده ای از کاراکترها استفاده شد.
- برای رشته طول متغیر با حد معین : نمایش حافظه از توصیفگری استفاده می کند که حاوی حداکثر طول و طول فعلی رشته ذخیره شده در شی داده است.
- برای رشته های نامحدود : می توان از نمایش حافظه پیوندی اشیا داده طول ثابت استفاده کرد

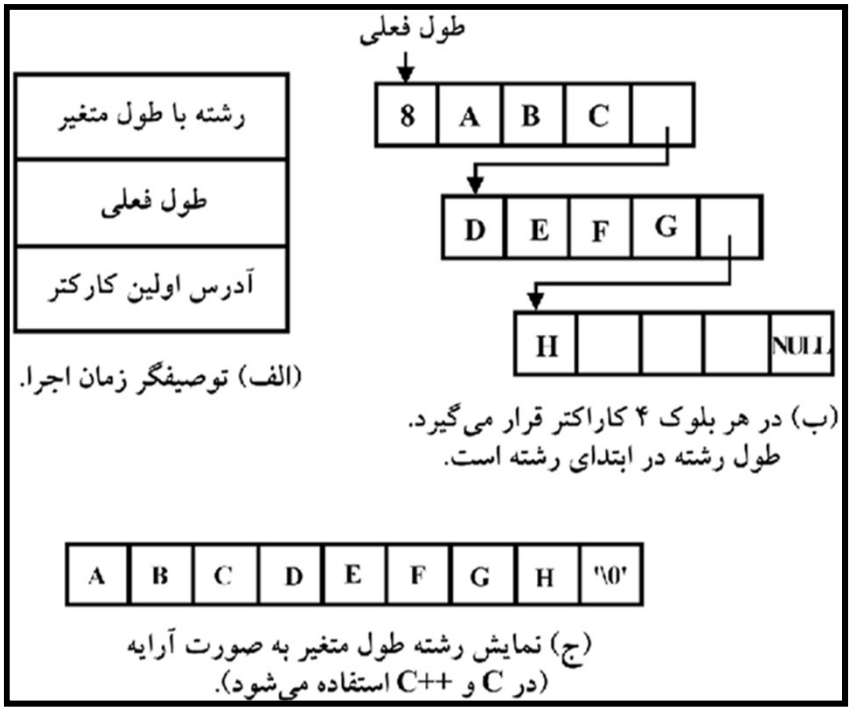
انواع داده مرکب(ادامه)

پیاده سازی رشته با طول ثابت.



(الف) توصیفگر زمان ترجمه.

(ب) اگر تعداد کاراکترها کمتر از طول رشته باشد بقیه خالی است.

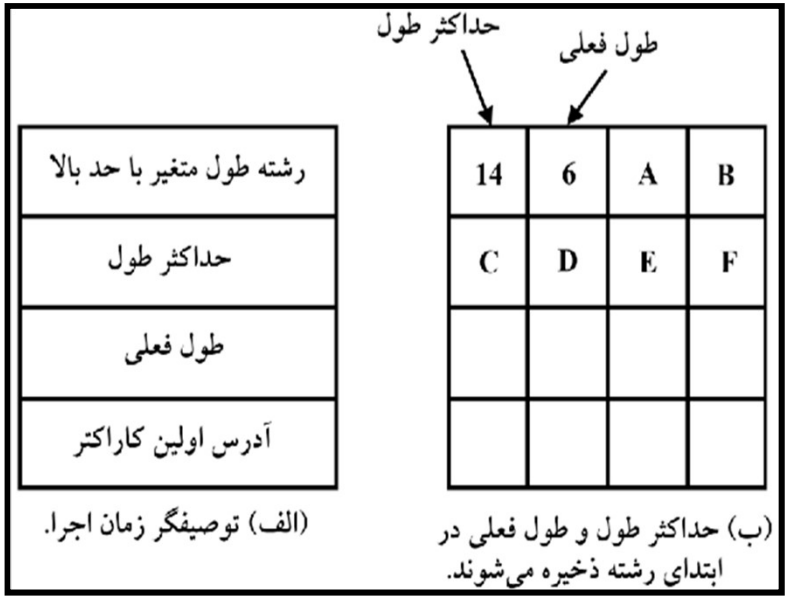


(الف) توصیفگر زمان اجرا.

(ب) در هر بلوک ۴ کاراکتر قرار می‌گیرد. طول رشته در ابتدای رشته است.

(ج) نمایش رشته طول متغیر به صورت آرایه (در C و C++ استفاده می‌شود).

پیاده سازی رشته با طول متغیر.



(الف) توصیفگر زمان اجرا.

(ب) حداکثر طول و طول فعلی در ابتدای رشته ذخیره می‌شوند.

پیاده سازی رشته با طول متغیر با حد بالا.

انواع داده مرکب(ادامه)

اشاره گرها و اشیای داده برنامه نویس

- زبان باید ویژگیهای زیر را داشته باشد:
  - نوع داده اولیه اشاره گر
  - عمل ایجاد کردن
  - عملیات دستیابی به محتویات

انواع داده مرکب(ادامه)

اشاره گرها و اشیای داده برنامه نویس (ادامه)

- مشخصات: نوع داده اشاره گر دسته از اشیای داده را تعریف می کند که مقادیر آنها آدرسهای اشیای دیگری اند
  - اشاره گر ها ممکن است فقط به یک نوع شی داده مراجعه کنند.
  - اشاره گرها ممکن است به هر نوع شی داده مراجعه کنند.

انواع داده مرکب(ادامه)

اشاره گرها و اشیای داده برنامه نویس (ادامه)

- پیاده سازی: شی داده اشاره گر به صورت محلی از حافظه نمایش داده می شود که شامل آدرس محل دیگری از حافظه است.
  - دو نمایش حافظه برای مقادیر اشاره گر استفاده می شود:
    - آدرس مطلق
    - آدرس نسبی

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی

- فایل ساختمان داده ای با دو ویژگی است:
  - بر روی حافظه ثانویه مثل دیسک یا نوار تشکیل می شود و ممکن است بسیار بزرگتر از سایر ساختمان داده ها باشد.
  - طول عمر آن می تواند بسیار زیاد باشد.

انواع داده مرکب(ادامه)

فایلها و ورودی – خروجی (ادامه)

- انواع فایل
  - متداول ترین فایلها، فایلهای ترتیبی اند.
  - فایلهای دستیابی مستقیم (direct access)
  - فایلهای ترتیبی شاخص دار (indexed sequential files)
  - فایلهای متنی

فصل چهارم

بسته بندی

ساختمان داده ها

اشیای داده ساختاری و انواع داده

- شی داده ای که مرکب از اشیای داده دیگری است ***ساختمان داده*** نام دارد.
- ساختمان داده های مهم عبارتند از آرایه ها، رکوردها، پشته ها، لیسها و مجموعه ها.
- انواع ساختمان داده نیز همانند انواع اولیه، اصولی مثل مشخصات، پیاده سازی، اعلانها و کنترل نوع دارند.

ساختمان داده ها (ادامه)

مشخصات انواع ساختمان داده

- صفات اصلی مشخص کننده ساختمان داده:
  - تعداد عناصر
    - اندازه ثابت
    - اندازه متغیر
  - نوع هر عنصر
    - همگن (Homogenous)
    - غیرهمگن (Hetergenous)
  - اسامی برای انتخاب عناصر
  - حداکثر تعداد عناصر
  - سازمان عناصر
    - ترتیبی
    - غیر ترتیبی

ساختمان داده ها (ادامه)

مشخصات انواع ساختمان داده (ادامه)

عملیات در ساختمان داده ها

- دسته های دیگری از عملیات از اهمیت ویژه ای برخوردارند:
  - عملیات انتخاب عناصر
    - انتخاب تصادفی
    - انتخاب ترتیبی
  - عملیات بر روی کل ساختمان
  - درج و حذف عناصر
  - ایجاد و حذف ساختمان داده ها

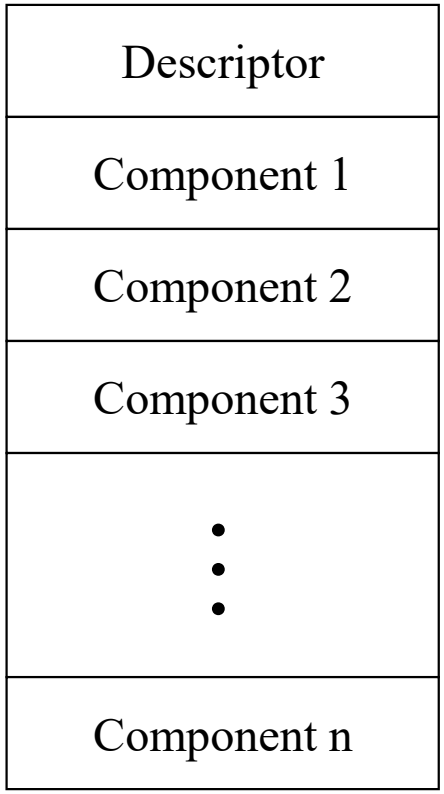
ساختمان داده ها (ادامه)

مشخصات انواع ساختمان داده (ادامه)

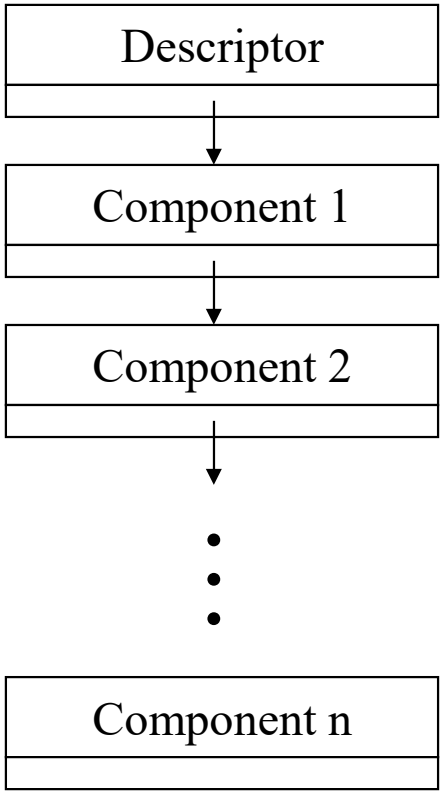
نمایش های حافظه

- نمایش حافظه برای ساختمان داده ها شامل:
  1. حافظه ای برای عناصر ساختمان داده
  2. توصیفگر (descriptor) اختیاری آنها
- دو نمایش اصلی:
  - نمایش ترتیبی (sequential)
  - نمایش پیوندی (linked)
- به عنان قاعده مکانیزم ترتیبی برای طول ثابت و مکانیزم پیوندی برای طول متغیر استفاده می شود.

ساختمان داده ها (ادامه)



Sequntial



Linked

ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها (ادامه)

پیاده سازی عملیات ساختمان داده ها

- انتخاب عناصر ساختمان داده مهمترین مسئله در پیاده سازی آن است
- کارآمد بودن عملیات انتخاب تصادفی و انتخاب ترتیبی ضروری است.

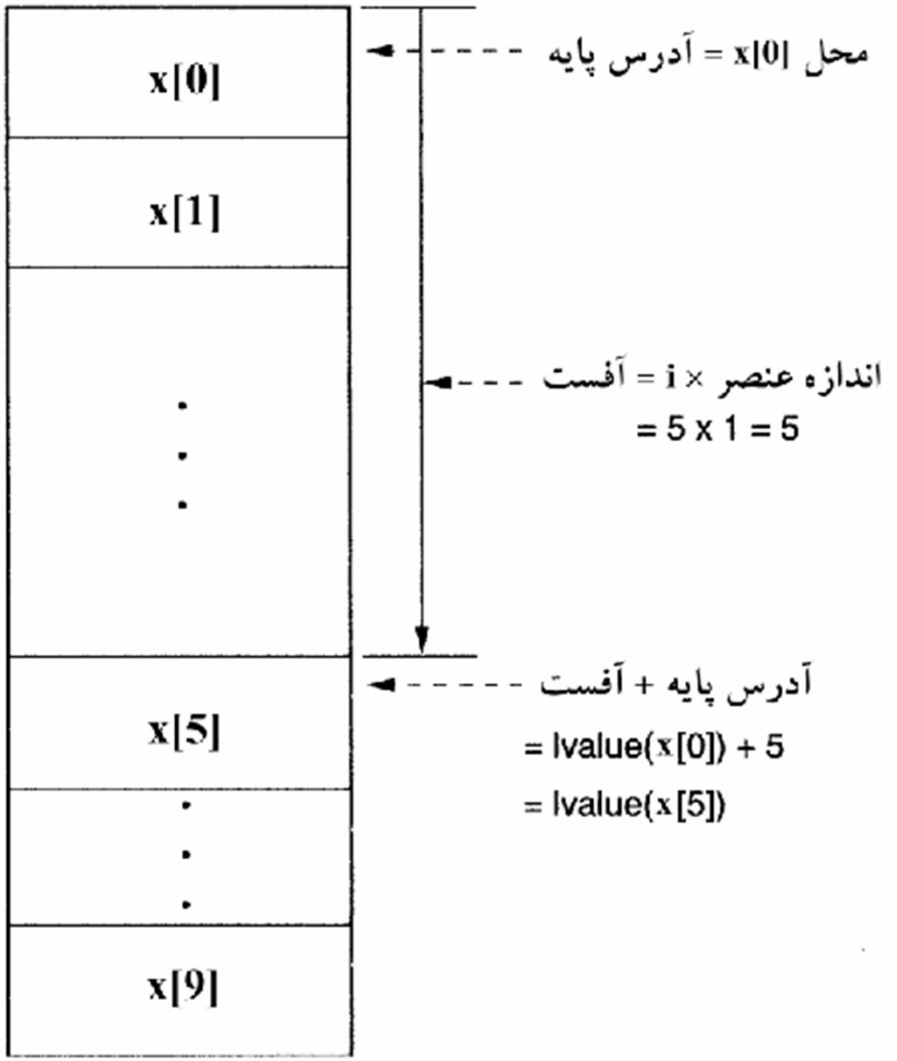
ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها (ادامه)

پیاده سازی عملیات ساختمان داده ها (ادامه)

- نمایش ترتیبی: در انتخاب تصادفی یک آدرس پایه – آفست باید با استفاده از فرمول دستیابی محاسبه شود.
  - محل شروع بلوک آدرس پایه نام دارد.
  - محل نسبی عنصر انتخاب شده در بلوک ترتیبی آفست نام دارد.
- در ساختار همگن انتخاب دنباله ای از عناصر می تواند به صورت زیر انجام شود:
  - برای دستیابی به اولین عنصر دنباله از محاسبه آدرس پایه – آفست استفاده کنید.
  - برای دستیابی به عنصر بعدی دنباله اندازه عنصر فعلی را به موقعیت عنصر فعلی اضافه کنید.

ساختمان داده ها (ادامه)



ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها (ادامه)

پیاده سازی عملیات ساختمان داده ها (ادامه)

- نمایش پیوندی: برای انتخاب باید زنجیره ای از اشاره گرها را از اولین بلوک موجود در ساختار تا عنصر مورد نظر دنبال کرد.
- برای انتخاب دنباله ای از مولفه ها باید اولین عنصر را انتخاب و سپس اشاره گر پیوندی را تا عنصر مورد نظر دنبال کرد.

ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها (ادامه)

مدیریت حافظه و ساختمان داده ها

- طول عمر هر شی داده با انقیاد شی به محلی از حافظه شروع می شود.
- وقتی انقیاد از بین رفت طول عمر آن از بین می رود.
- به علت تاثیر متقابل بین طول عمر شی داده و مسیرهای دستیابی دو مسئله مهم در مدیریت حافظه به وجود می آید:
  - زباله (garbage)
  - ارجاعهای معلق (dandling reference)

ساختمان داده ها (ادامه)

اعلانها و کنترل نوع برای ساختمان داده ها

- مثل انواع داده اولیه است
- ساختمان داده ها پیچیده ترند زیرا صفات بیشتری باید مشخص شوند.
- دو مسئله در این مورد وجود دارد:
  - وجود مولفه انتخابی
  - نوع عنصر انتخابی

ساختمان داده ها (ادامه)

بردارها و آرایه ها

- متداولترین ساختمان داده ها در زبانهای برنامه سازی اند.
- بردار ساختمان مرکب از تعداد ثابتی از عناصر همنوع است که به صورت یک دنباله خطی سازمان دهی شده است.
- برای دستیابی به عناصر بردار از اندیس استفاده می شود.
  - اندیس یک مقدار صحیح یا شمارشی است که موقعیت یک عنصر را در بردار مشخص می کند.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

بردارها:

- تعداد عناصر
- نوع هر عنصر
- اندیس برای انتخاب هر عنصر

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

عملیات بر روی بردارها:

- عملیاتی که عنصری را از برداری انتخاب می کند اندیس گذاری نام دارد.
  - عمل اندیس گذاری، مقدار چپ یا محل شئ داده را برمی گرداند.
- عملیات دیگر بردار عبارتند از ایجاد و از بین بردن آنها، انتساب به عناصری از بردار و عملیاتی که اعمال محاسباتی را بر روی بردارهایی با طول یکسان انجام می دهند، مانند جمع دو بردار.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی:

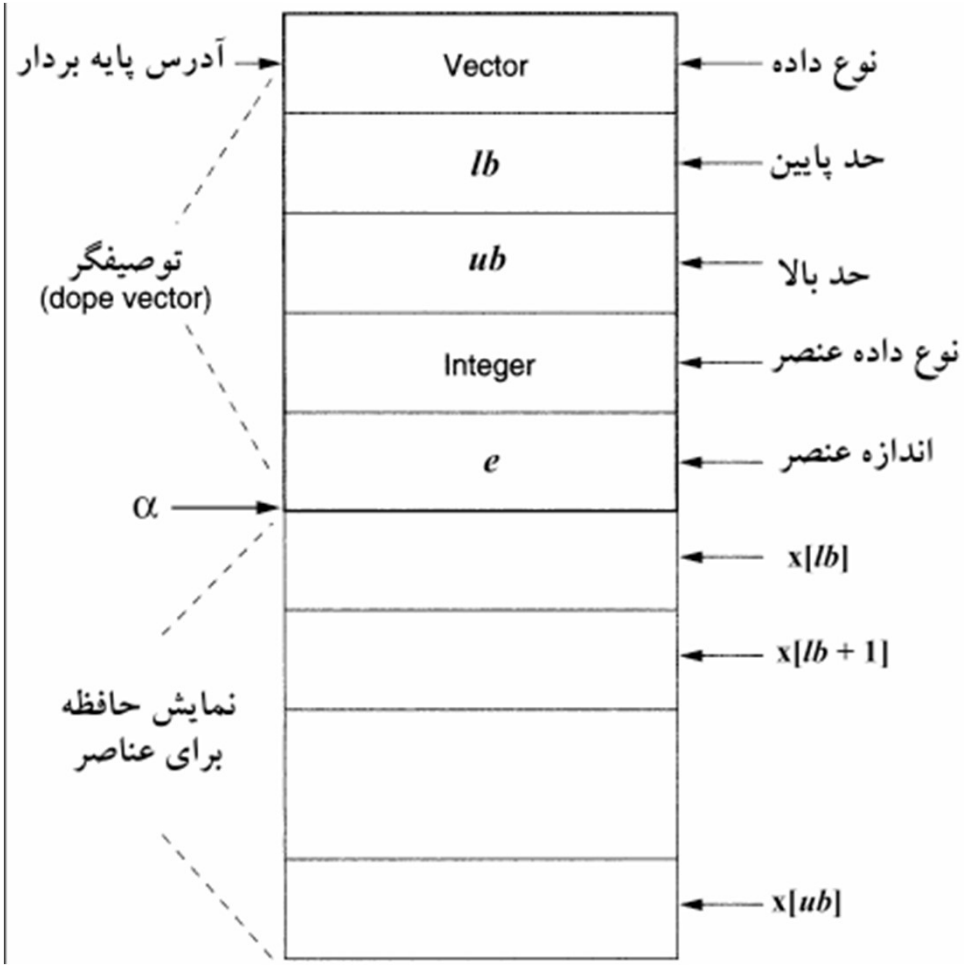
- به دلیل همگنی عناصر، اندازه و ساختار عناصر یکسان است.
- به دلیل طول ثابت، تعداد و موقعیت هر عنصر بردار در طول عمرش ثابت است.
- نمایش حافظه ترتیبی مفید است.
- برای ذخیره صفات بردار می توان از توصیفگر استفاده کرد.
  - مخصوصاً اگر این اطلاعات تا زمان اجرا مشخص نباشد.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی (ادامه):

- نمایش حافظه برداری با توصیفگر کامل.



ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی (ادامه):

- اگر اولین عنصر در محل α باشد، مقدار چپ یک عنصر بردار عبات است از:

$$Lvalue(A[i]) = \alpha + (i-LB) \times E = (\alpha - LB \times E) + (i \times E)$$

- اگر $(\alpha - LB \times E)$ را K در نظر بگیریم خواهیم داشت.

$$Lvalue(A[i]) = K + i \times E$$

- فرمول دستیابی در C:

$$Lvalue(A[i]) = \alpha + i$$

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی (ادامه):

- مبدأ مجازی:
  - آدرس عنصری با اندیس صفر را مشخص می کند.

$$\begin{aligned} Lvalue(A[0]) &= (\alpha - LB \times E) + (0 \times E) \\ &= (\alpha - LB \times E) \\ &= K \end{aligned}$$

  - K آدرس عنصر صفر بردار (در صورت وجود) می باشد.
- چون ممکن است عنصر صفر وجود نداشته باشد این آدرس، مبدأ مجازی (VO) است.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی (ادامه):

- الگوریتم ساختن بردار و تولید فرمول دستیابی:
  1. هنگام تخصیص حافظه برای بردار.
     - $D + N \times E$ محل حافظه را تخصیص بده.
     - آدرس اولین عنصر را در α قرار بده.
  2. مبدأ مجازی را محاسبه کن : $VO = \alpha - LB \times E$
  3. در دستیابی به عنصر بردار:

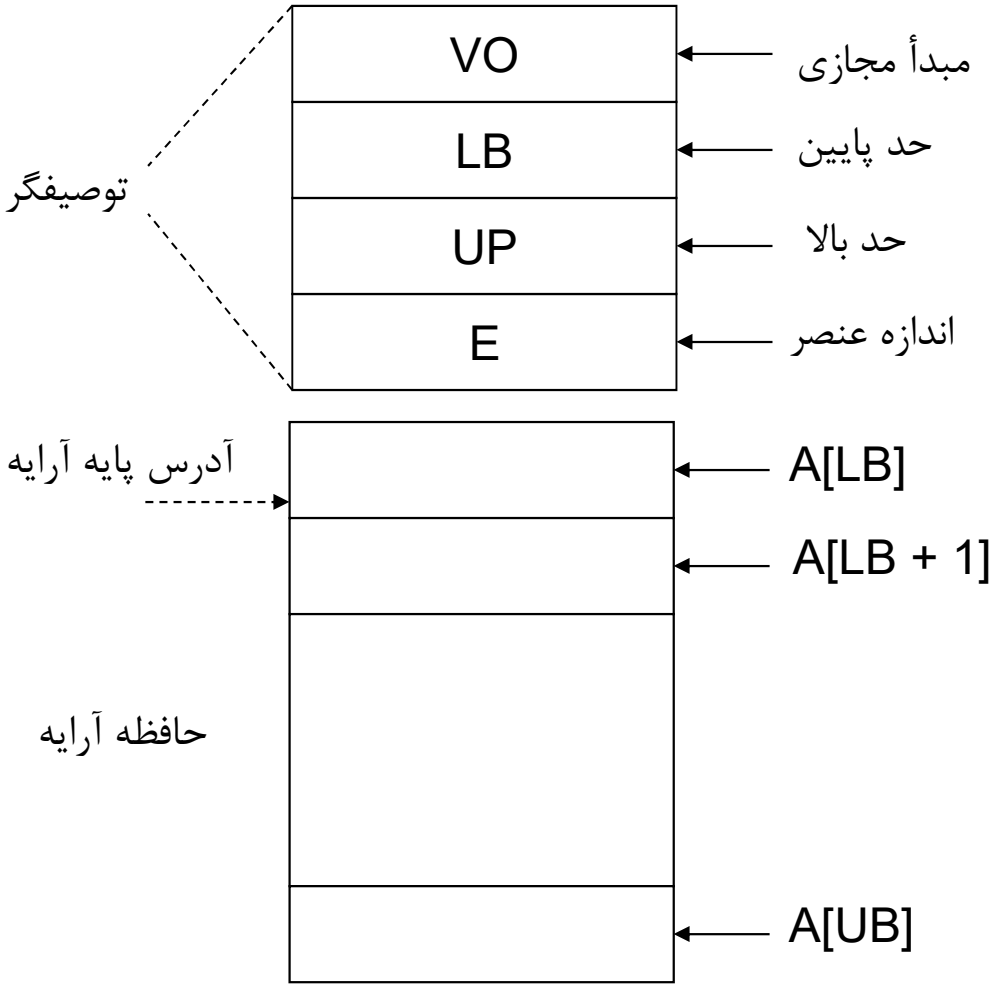
$$Lvalue(A[i]) = VO + i \times E$$

- اگر مبدأ مجازی در توصیف گر آرایه ذخیره شود، آرایه واقعی لازم نیست با توصیفگرش به طور پیوسته قرار داشته باشد.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی (ادامه):



ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی (ادامه):

- توصیفگرهای مربوط به پارامترهای آرایه می تواند به زیربرنامه ها ارسال شود ولی آرایه واقعی در جای دیگری ذخیره شده باشد.
  - نمایشهای حافظه به صورت فشرده و غیرفشرده
  - عملیات بر روی کل بردار

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

آرایه های چند بعدی

- مشخصات و نحو: تفاوت آرایه چند بعدی و بردار در بازه اندیس هر بعد است.
- پیاده سازی: می توان آن را برداری از بردارها در نظر گرفت .
  - آرایه ها با ابعاد بیشتر را می توان با آرایه هایی با ابعاد کمتر ایجاد کرد.
  - تعداد عناصر تمام بردارهای فرعی (subvector) و نوع هر عنصر باید یکسان باشد.
  - آرایه می تواند column major یا row major باشد.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

آرایه های چند بعدی

- فرمول دستیابی به صورت زیر خواهد بود

$$Lvalue(A[i,j]) = \alpha + (i - LB_1) \times S + (j - LB_2) \times E$$

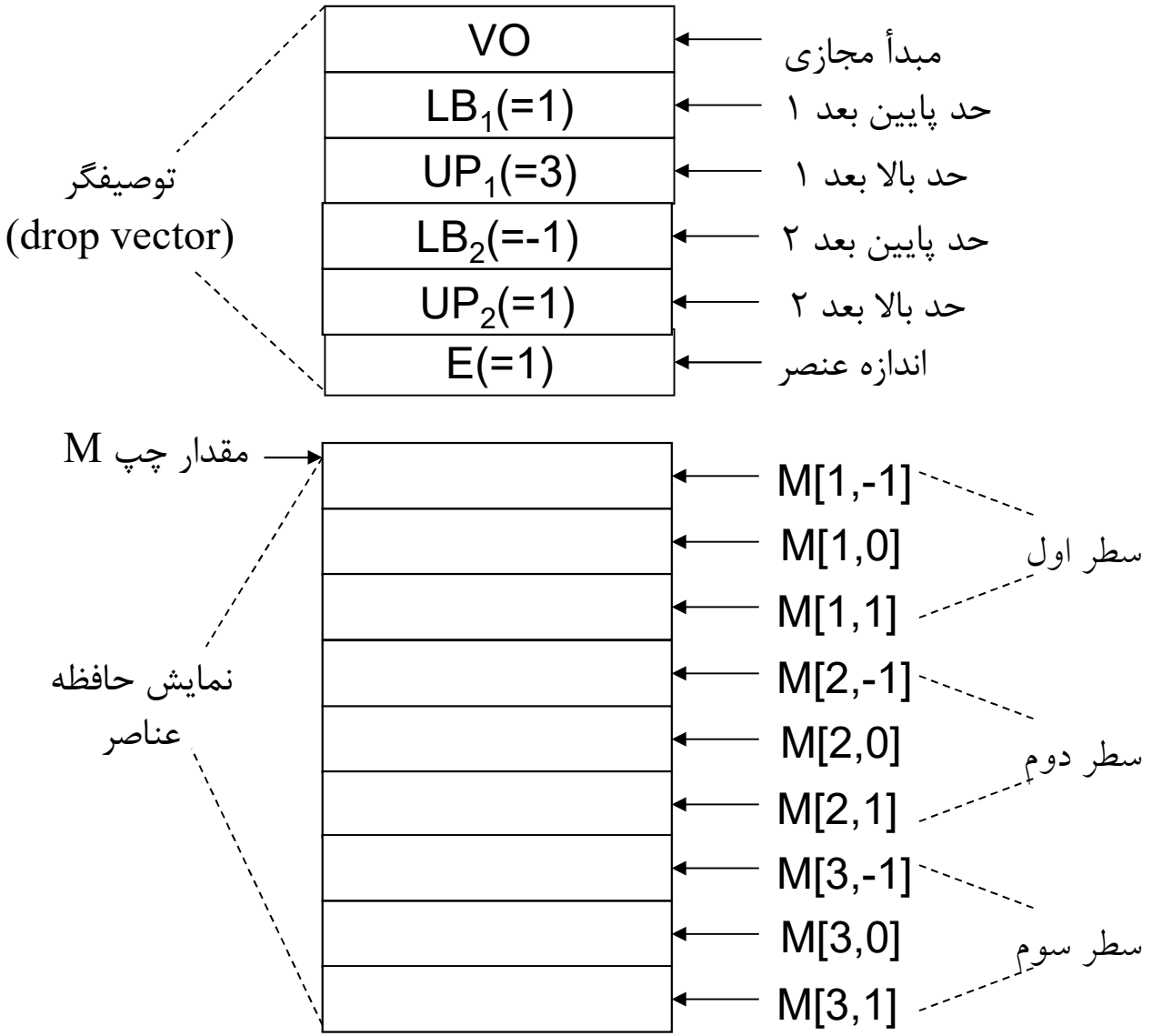
- بنابر این خواهیم داشت

$$S = (UB_2 - LB_2 + 1) \times E$$

$$VO = \alpha - LB_1 \times S + LB_2 \times E$$

$$Lvalue(A[i,j]) = VO + i \times S + j \times E$$

ساختمان داده ها (ادامه)

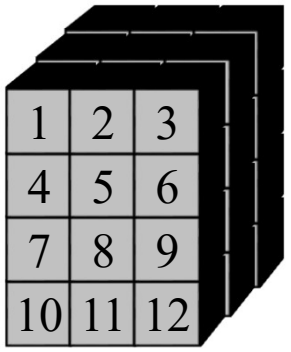


ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

برش آرایه (slice)

- مشخصات : برش بخشی از آرایه است که خودش یک آرایه است.
- پیاده سازی: استفاده از توصیفگر منجر به پیاده سازی کارآمد برشها می شود.



برش چند بعدی

1	2	3
4	5	6
7	8	9
10	11	12

برش سطری

1	2	3
4	5	6
7	8	9
10	11	12

برش ستونی

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

آرایه های شرکت پذیر (associative arrays)

- از طریق نام بتوان به اطلاعات دست یافت.
- از نام بعنوان اندیس استفاده شود.
- به دلیل نا محدود بودن دنباله ای از اسامی، استفاده از نوع داده ای شمارشی امکانپذیر نیست.
- مجموعه ای از اسامی به عنوان مجموعه شمارشی بکار گرفته می شود.
- اگر نام جدیدی اضافه شود این شمارشگر افزایش می یابد.

ساختمان داده ها (ادامه)

رکوردها

- رکورد ساختان داده ای مرکب از تعداد ثابتی از عناصر و از انواع مختلف است.
- مشخصات و نحو: ساختمان داده های خطی با طول ثابت هستند اما رکوردها از دو جهت متفاوتند:
  - عناصر رکورد ممکن است ناهمگن و از انواع مختلفی باشند.
  - عناصر رکورد توسط اسامی symboic مورد دسترسی قرار می گیرند.
- صفات رکورد عبارت است از:
  - تعداد عناصر (فیلد)
  - نوع هر عنصر
  - نامگذاری هر عنصر
- رکورد را ساختمان نیز می نامند.

ساختمان داده ها (ادامه)

رکوردها (ادامه)

- عملیات:
  - عملیات روی هر عنصر
    - بستگی به نوع هر عنصر دارد.
  - عملیات روی کل ساختار
    - محدود و تنها assignment یک کورد روی رکورد دیگر است.
- پیاده سازی: نمایش حافظه برای رکورد شامل یک بلوک از حافظه است که عناصر درآن به ترتیب ذخیره می شود.
  - معمولاً descriptor زمان اجرا در نظر گرفته نمی شود.
  - مکان هر مولفه به شکل زیر آدرس دهی می شود.

$$Lvalue(R.I) = \alpha + \sum_{J=1}^{I-1} (size \quad of \quad R.J)$$

ساختمان داده ها (ادامه)

رکوردها (ادامه)

- مثال:

```
struct student {
     int id;
    int age;
    float ave;
    char name[20];
}
studentd st, sp;
```

id	age	ave	name
100	18	15.5	Ali

نمایش حافظه رکورد student.

ساختمان داده ها (ادامه)

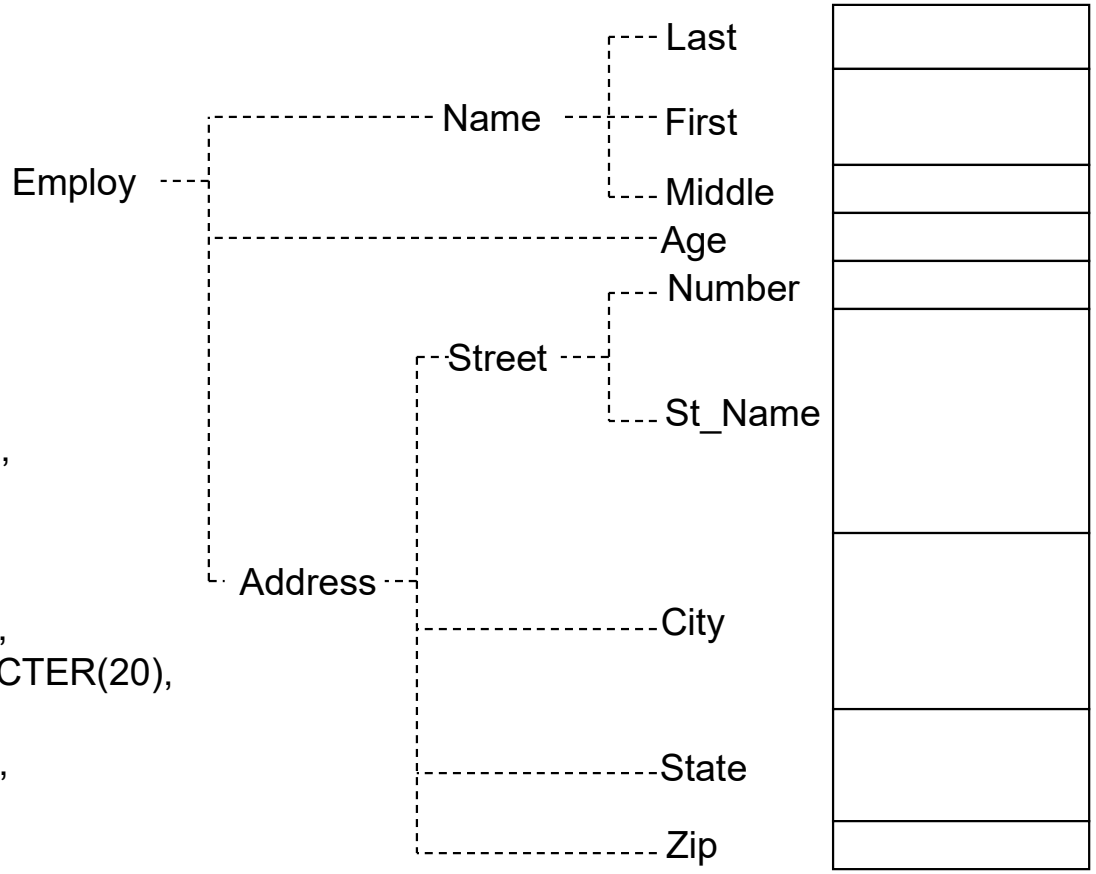
رکوردها (ادامه)

رکوردها و آرایه هایی با عناصر ساختاری

- عناصری از دو نوع مختلف با عناصری از انواع داده ترکیب شوند.
- انتخاب عناصر مستلزم دنباله ای از انتخابها با شروع از آدرس پایه ساختمان اصلی و محاسبه یک آفست برای یافتن محل عنصر اولین سطح و سپس محاسبه یک آفست از این آدرس پایه برای یافتن عناصر دومین سطح و غیره است.

ساختمان داده ها (ادامه)

```
1 Employ,
        2 Name,
                3 Last CHARACTER(10),
                3 First CHARACTER(15),
                3 Middle CHARACTER(1),
        2 Age FIXED(2),
        2 Address
                3 Street,
                        4 Number FIXED(5),
                        4 St_Name CHARACTER(20),
                3 City CHARACTER(15),
                3 State CHARACTER(10),
                3 Zip FIXED(5);
```



ساختمان داده ها (ادامه)

رکوردها (ادامه)

رکوردهای طول متغیر

- نوعی است که در زمان های مختلف اجرای برنامه، ساختار های متفاوتی دارد.
- در رکوردهای طول متغیر عناصر ممکن است در یک زمان وجود داشته باشند و در زمان دیگر وجود نداشته باشند.
- منظور این است که معتبر بودن یا نبودن یک عنصر بستگی به ارزش عنصر دیگر داشته باشد. برای حل مشکل:
  - کنترل پویا (Dynamic checking)
  - کنترلی انجام نشود (No checking).

ساختمان داده ها (ادامه)

رکوردها (ادامه)

رکوردهای طول متغیر (ادامه)

- پیاده سازی : با توجه به اعلان رکورد طول متغیر، مترجم اندازه حافظه مربوط به ساختارهای مختلف رکورد را محاسبه کرده، حافظه ای به اندازه بزرگ ترین ساختار ممکن تخصیص می یابد، به طوری که آدرس تمام فیلدهای متغیر از یک نقطه شروع می شود.
- توصیف کاملی از هر ساختار رکورد طول متغیر باید در زمان ترجمه وجود داشته باشد.

ساختمان داده ها (ادامه)

رکوردها (ادامه)

رکوردهای طول متغیر (ادامه)

```
Type PayType = (Salaried, Hourly);
Var Employee: record
            ID: integer;
            Dept: array [1..3] of char;
            Age: integer;
            case PayClass: PayType of
              Salaried: (MnthlyRate: real;
                         StartDate: integer);
              Hourly: (HourRate: real;
                       Reg: integer;
                       Overtime: integer)
            end
```

ID	
Dept	
Age	
PayClass	
MonthlyRate	HourRate
StartDate	Reg
	Overtime

ساختمان داده ها (ادامه)

لیست ها

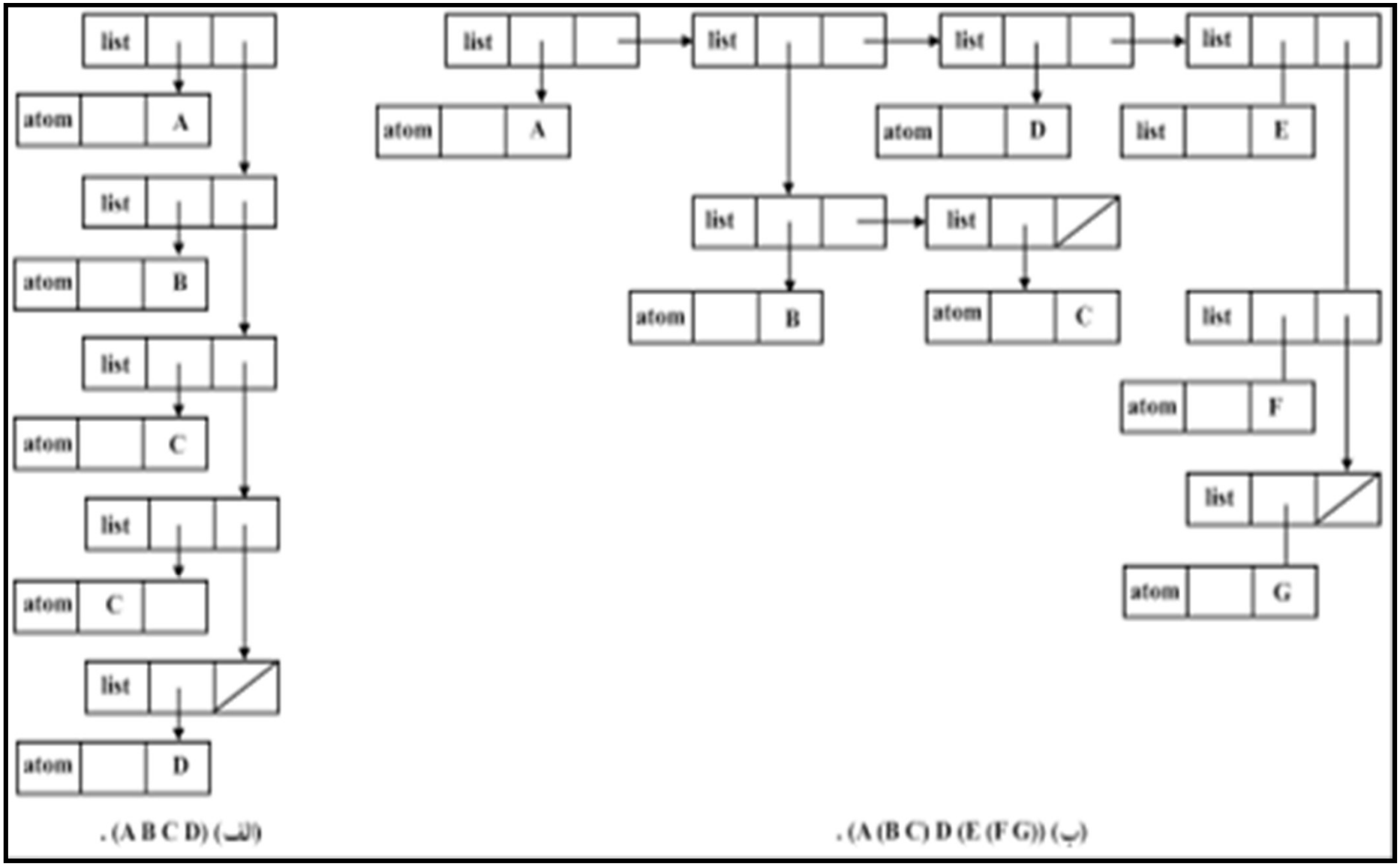
- مشخصات و نحو: لیستها همانند بردارها حاوی دنباله مرتبی از اشیا هستند.
- اولین عنصر لیست را معمولاً راس می گویند.
- لیستها از جهات زیر با بردارها متفاوتند:
  - طول لیست به ندرت ثابت است
  - لیستها به ندرت همگن هستند.
- زبانهایی که از لیستها استفاده می کنند، نوعاً چنین داده ای را بصورت ضمنی، بدون صفات صریح برای اعضای لیست، تعریف می کنند.

ساختمان داده ها (ادامه)

لیست ها(ادامه)

- پیاده سازی: مدیریت حافظه منظم که برای بردارها و آرایه ها مفید است در اینجا قابل استفاده نیست.
- معمولاً از سازمان مدیریت حافظه پیوندی استفاده می شود.
- قلم لیست یک عنصر اولیه است که معمولاً شامل شی داده ای به اندازه ثابت است.
- لیسپ سه فلید اطلاعات نیاز دارد:
  - یک فیلد نوع
  - دو فیلد اشاره گر لیست

ساختمان داده ها (ادامه)



ساختمان داده ها (ادامه)

لیست ها (ادامه)

- شکلهای گوناگون لیستها:
  - پشته ها و صفها
  - درختها
  - گرافهای جهت دار
  - لیستهای خاصیت

ساختمان داده ها (ادامه)

مجموعه ها

- مجموعه شی داده ای است که شامل مقادیر نامرتب و مجزا است.
- عملیات اصلی روی مجموعه ها عبارتند از:
  - عضویت
  - درج و حذف یک مقدار
  - اجتماع، اشتراک و تفاضل مجموعه ها

ساختمان داده ها (ادامه)

مجموعه ها(ادامه)

- پیاده سازی:
  - مجموعه ساختمان داده ای است که عناصر مرتب را نشان می دهد.
  - مجموعه مرتب لیستی است که مقادیر تکراری آن حذف شده اند.
  - مجموعه نامرتب دو نمایش حافظه دارد
    - نمایش بیتی مجموعه ها
    - نمایش درهم سازی مجموعه ها

انواع داده انتزاعی

تکامل مفهوم نوع داده

- مفهوم اولیه نوع داده نوع را به صورت مجموعه ای از مقادیر تعریف می کند که یک متغیر می تواند آنها را بپذیرد.
- نمایش حافظه مربوط به مقادیر حقیقی و صحیح کاملاً بسته بندی شده است یعنی از برنامه نویس پنهان است.
- برنامه نویس بدون اینکه از جزئیات نمایش حافظه این انواع اطلاع داشته باشد از اشیای داده آنها استفاده می کند.
- برنامه نویس فقط نام نوع و عملیاتی را برای دستکاری آن نوع فراهم می بیند.

انواع داده انتزاعی(ادامه)

تکامل مفهوم نوع داده

انتزاع داده ها

- نوع داده انتزاعی :
  - مجموعه ای از اشیای داده معمولاً با استفاده از یک یا چند تعریف نوع
  - مجموعه ای از عملیات انتزاعی بر روی آن انواع داده
  - بسته بندی تمام آنها به طوری که کاربر نوع جدید نتواند اشیای داده ای از آن نوع را به جز از طریق عملیاتی که برای آن تعریف شده است دستکاری کند.

انواع داده انتزاعی (ادامه)

پنهان سازی اطلاعات

- برای نوشتن برنامه بزرگ باید از استراتژی تقسیم و حل استفاده کرد
  - برنامه به مجموعه ای از قطعات، به نام ماژول (module) تقسیم می شود.
- طراحی ماژول معمولاً به دو روش انجام می شود:
  - ماژولها، تجزیه تابعی برنامه را نشان می دهند.
  - ماژولها، تجزیه داده ای برنامه را نشان می دهند.

انواع داده انتزاعی (ادامه)

پنهان سازی اطلاعات(ادامه)

- زبان برنامه سازی انتزاع را به دو روش پشتیبانی می کند:
  - با تدارک کامپیوتر مجازی که کاربرد آن ساده تر و قدرت آن بیش از کامپیوتر سخت افزار است.
  - زبان امکاناتی را فراهم می کند که برنامه نویس می تواند انتزاعها را به وجود آورد.
- بسته بندی اصلاح برنامه را آسان می کند.
- زیر برنامه ها مکانیزم بسته بندی را شکل می دهند که تقریباً در هر زبانی وجود دارد.

بسته بندی با زیربرنامه ها(ادامه)

- زیر برنامه ها و عملیات انتزاعی
- مشخصات زیربرنامه:
  - نام
  - امضای زیربرنامه
  - فعالیتی که توسط زیربرنامه انجام می شود.
- بیان دقیق زیربرنامه ها به صورت تابع ریاضی با مشکلات ریاضی روبرو است:
  - زیربرنامه ممکن است پارامتر های ضمنی به صورت متغیر های غیر محلی داشته باشند.
  - زیربرنامه ممکن است اثرات جانبی داشه باشد(مثل تغییر در پارامتر ها).
  - زیربرنامه ممکن است برای بعضی از مقادیر دامنه (پارامتر ها) تعریف نشده باشد.
  - زیربرنامه ممکن است به سابقه اجرا حساس باشد(وجود متغیر های استاتیک).
- پیاده سازی زیر برنامه
  - با استفاده از ساختمان داده ها و عملیاتی پیاده سازی می شود که توسط خود زبان برنامه نویسی ارائه می شود.

بسته بندی با زیربرنامه ها (ادامه)

تعریف و فراخوانی زیربرنامه

- تعریف زیر برنامه خاصیت ایستای یک برنامه است.
- درحین اجرای برنامه اگر زیر برنامه ای فراخوانی شود سابقه فعالیتی از آن زیر برنامه ایجاد می شود. و وقتی زیر برنامه خاتمه یافت، سابقه آن نیز از بین می رود
- تعریف زیربرنامه قالبی (template) برای ایجاد سابقه فعالیت در حین اجرا است.
- شی داده در حین اجرا برنامه ایجاد می شود:
  - در حین ورود به زیر برنامه
  - توسط عملیاتی مثل malloc

بسته بندی با زیربرنامه ها (ادامه)

تعریف و فراخوانی زیربرنامه(ادامه)

پیاده سازی تعریف و فراخوانی زیربرنامه

- برای ساختن سابقه فعالیت خاصی از الگوی زیر برنامه، کل الگوی باید در ناحیه جدیدی از حافظه کپی شود.
- اما به جای کپی کامل بهتر است الگو به دو بخش تقسیم شود:
  - بخش ایستا که سگمنت کد نام دارد و حاوی ثوابت و کد اجرایی است.
  - بخش پویا که رکورد فعالیت نام دارد و شامل پارامترها، نتایج تابع، داده های محلی، ناحیه،حافظه موقت، نقاط برگشت، و پیوند هایی برای مراجعه به متغیر های غیر محلی است.

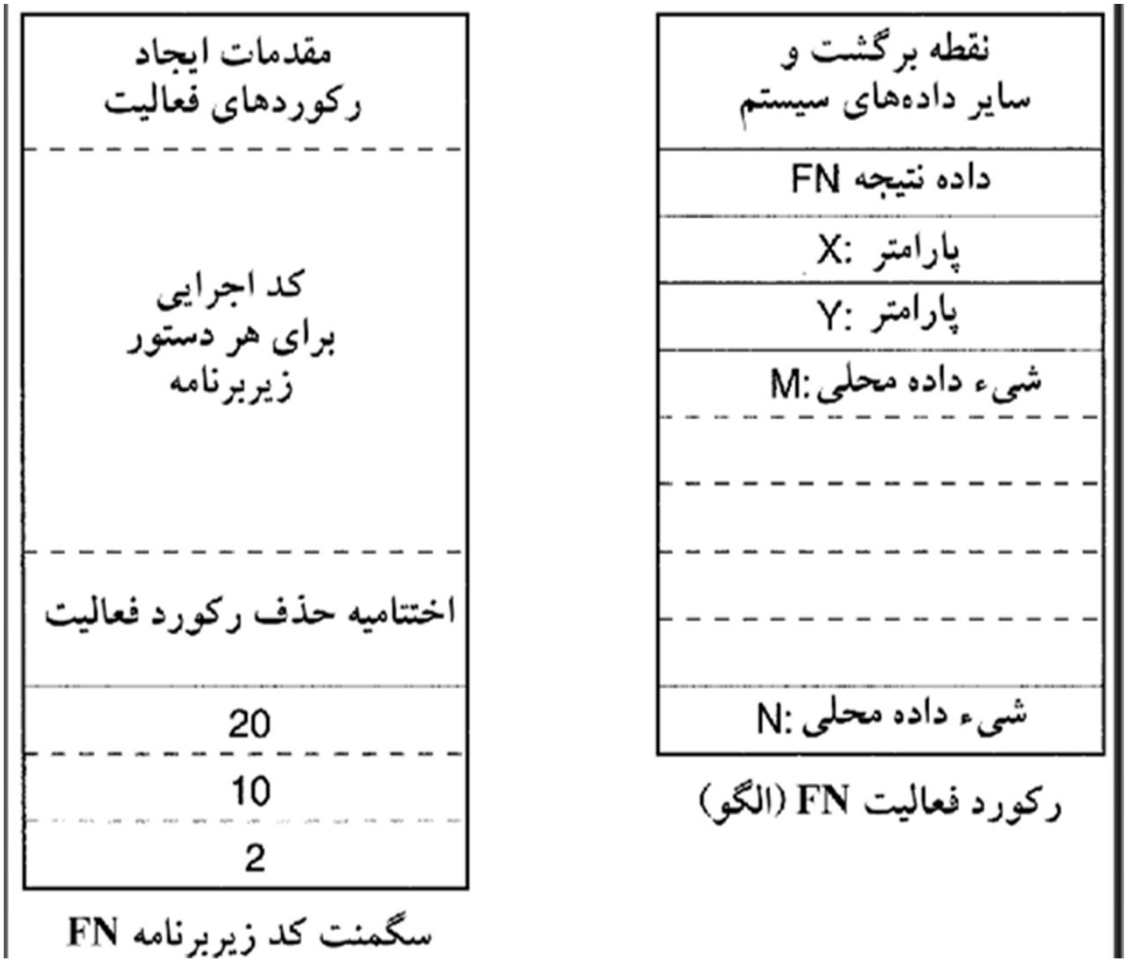
بسته بندی با زیربرنامه ها (ادامه)

تعریف زیربرنامه به عنوان اشیای داده

- ترجمه عملیاتی است که تعریف زیربرنامه را به شکل رشته کاراکتری گرفته شی داده زمان اجرا را تولید می کند که این تعریف را نمایش می دهد.
- اجرا عملیاتی است که تعریفی به شکل زمان اجرا را گرفته سابقه فعالیتی را از آن ایجاد می کند و آن سابقه فعالیت را اجرا می نماید.

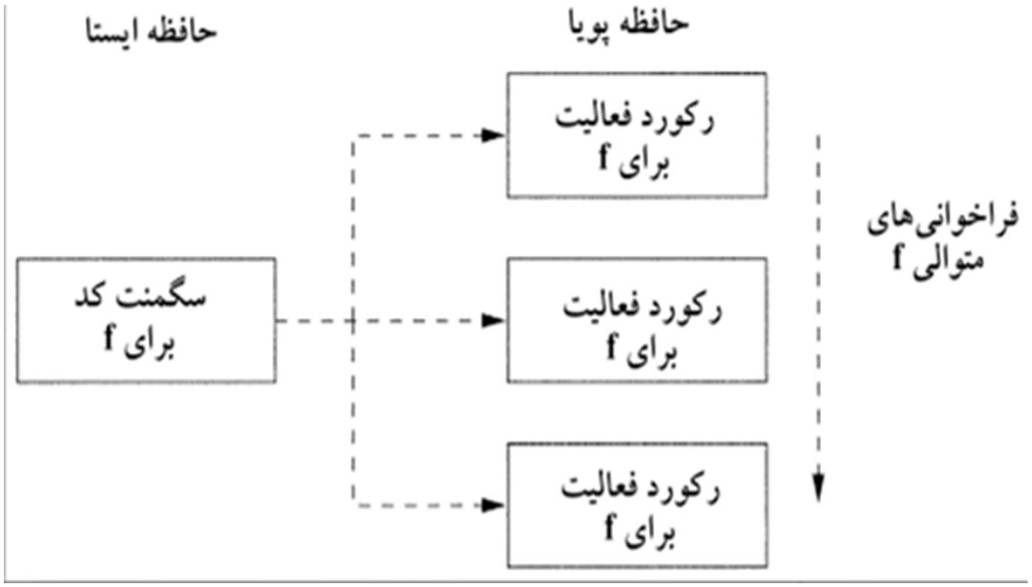
بسته بندی با زیربرنامه ها (ادامه)

```
float f(float x ,int y)
{
    const int value = 2;
    float m[20];
    int n;
    .
    .
    .
    return(10 * x + m[n])
}
```



سابقه فعالیت زیر برنامه.

بسته بندی با زیربرنامه ها (ادامه)



چندرکورد فعالیت با سگمنت کد مشترک.

تعریف نوع

- در اغلب زبان های جدید، امکان تعریف نوع گنجانده شده است.
- تعریف زیر را در C ببینید:

```
struct rational
{
    int num1;
    int num2;
};
```

- پیاده سازی: اطلاعات موجود در اعلان متغیرها در زمان ترجمه برای تعیین نمایش حافظه اشیا و اهداف مدیریت حافظه و کنترل نوع بکار می رود. اعلان ها در زمان اجرا و جود ندارند، بلکه برای ایجاد مناسب اشیا در زمان اجرا به کار می روند.
مترجم زبان، اطلاعات حاصل از تعریف نوع را در جدولی قرار می دهد و با استفاده از این اطلاعات، کد اجرایی مناسب را برای تنظیم و دستکاری اشیای مطلوب تولید می کند.

تعریف نوع(ادامه)

هم ارزی نوع

- دو راه حل برای مشخص کردن تساوی نوع وجود دارد
  - هم ارزی نام
    - معایب:
      - هر شی که در انتساب بکار می رود باید دارای نام باشد
      - یک تعریف نوع باید در سراسر برنامه یا بخش بزرگی از برنامه قابل استفاده باشد
  - هم ارزی ساختاری
    - معایب
      - آیا نام فیلدها، ترتیب فیلدها باید یکی باشد ...
      - دو متغیر ممکن است به طور تصادفی از نظر ساختاری یکسان باشند. بسیاری از خطا های نوع ممکن است تشخیص داده نشود.
      - تعیین هم ارزی ساختاری در مورد انواع پیچیده هزینه ترجمه دارد.

فصل پنجم

وراثت

نگاهی دوباره به انواع داده انتزاعی

- نوع داده انتزاعی نوع داده جدیدی است که توسط برنامه نویس تعریف می شود.
- داده انتزاعی شامل موارد زیر است:
  - نوع داده ای که توسط برنامه نویس تعریف شد.
  - مجموعه ای از عملیات انتزاعی بر روی اشیایی از آن نوع
  - بسته بندی اشیای آن نوع به طوریکه کاربر آن نوع نمی تواند آن اشیا را بدون استفاده از این عملیات دستکاری کند.

نگاهی دوباره به انواع داده انتزاعی

- انتزاع داده : طراحی اشیا داده و عملیات انتزاعی بر روی آن اشیا
- هر زیر برنامه ای که می تواند متغیری را از نوع جدید اعلان کند اجازه دارد به هر عنصر از نمایش آن نوع دستیابی داشته باشد.
- پیاده سازی: پکیج بسته بندی را برای تعریف نوع و زیربرنامه فراهم می کند.
- اولین اثرش محدود کردن قابلیت مشاهده اسامی اعلان شده در پکیج است.
- هر پکیج شامل دو بخش است:
  - مشخصات
  - پیاده سازی

نگاهی دوباره به انواع داده انتزاعی

انواع داده انتزاعی کلی:

- با استفاده از انواع داده اولیه ای که در زبان وجود دارند می تواند نوع پایه ای را برای دسته جدید از اشیا داده اعلان کند
- تعریف نوع انتزاعی کلی امکان صفت از یک نوع به طور جداگانه را فراهم می کند

کلاس در C++

- C++ علاوه بر ویژگی زبان دستوری C،یک زبان برنامه سازی شیء گرا نیز هست.نوع داده انتزاعی در C++ با استفاده از کلاس ها ایجاد می شوند . شکل کلی کلاس در C++ به صورت زیر است:

```
class نام کلاس {
  Private:
      داده ها و توابع اختصاصی
  Public:
      داده ها و توابه عمومی
  Protected:
      داده های محافظت شده
};
```

- نمونه ای از کلاس در C++ به صورت زیر است:

```
class circle {
    int radius;
    float area;
    float perime;
public:
    circle();
    ~circle();
    void get_radius();
    void calculator();
    void display();
};
```

```
class Stack
{
   int *x;
   int myTop;
   int max;
public:
   stack(int);
   int empty();
   int full();
   void push(int a);
   void pop();
   int top();
}; // end of class
Stack :: stack (int n)
{
   myTop = -1;
   max = n;
   x = new int{n};
}
void stack :: empty()
{
   return myTop == -1;
}
void stack :: full()
{
   return myTop >= max;
}
```

```
void stack :: push (int a)
{
    if(full())
        return;
    x{++myTop} = a;
}
void stack :: pop()
{
   if(!empty())
       myTop--;
}
int stack :: top()
{
   if(empty())
       return -1;
   return x[myTop];
}
int main()
{
   stack s1(10);
   stack s2(20);
   s1 .push(5);
   s2 .push(10);
   …
}
```

کلاس در C++ (ادامه)

- کامپایلر با دیدن تعریف یک کلاس، آن را به صورت زیر به تعریف ساختمان تبدیل می کند:

```
struct نام کلاس {
فیلد های داده
};ای
```

- بنابر این، کلاس stack به صورت زیر به ساختمان تبدیل می گردد:

```
struct stack {
    int *x;
    int myTop;
    int max;
};
```

- دراین صورت، وقتی s1 و s2 از نوع stack اعلان می شوند، درواقع به صورت دو ساختمان اعلان می شوند که هر کدام دارای فیلد هایی به نام های x ، myTop و max هستند.

کلاس در C++ (ادامه)

- توابع عضو کلاس (متدها) به تابعی در C تبدیل می شوند. به عنوان مثال، فرض کنید متد fعضو کلاس myClass باشد و به صورت زیر اعلان شده باشد:

```
type1 f(type2 p);
```

- یعنی تابع f از نوع type1 است و پارامتری به نام p دارد که از نوع type2 است. در این صورت، این متد به صورت زیر در می آید:

```
type1 myClassf (struct myClass *this, type2 p);
```

- یعنی نام کلاس به ابتدای نام تابع الحاق می شود و پارامتری به نام thisبه پارامترهای تابع اضافه می گردد. این پارامتر موجب می شود اشاره گر به ساختمانی که از کلاس به دست آمده است، به این تابع ارسال شود و این تابع از طریق اشاره گر، بتواند در آن تغییراتی ایجاد کند.

کلاس در C++ (ادامه)

- بار دیگر دو اعلان زیر را در برنامه در نظر بگیرید:

```
stack s1(10);
stack s2(20);
```

- این اعلان ها به کد زیر تبدیل می شوند:

```
struct stack s1;
stackstack(&s1, 10);
struct stack s2;
stackstack(&s2, 20);
```

- معنایش این است که با اعلان stack s1(n) تابع stack (سازنده پشته) به صورت زیر در می آید:

```
void stackstack(struct stack *this, int n)
{
   this -> myTop = -1;
   this -> max = n;
   this -> x = malloc(n *sizeof(int));
}
```

انواع داده انتزاعی کلی

- کلاس های کلی آن هایی هستند که نوع عناصر شان در زمان ایجاد اشیایی از آن کلاس مشخص می گردد. بر اساس نوعی که هنگام ایجاد شی ء تعیین می شود، کد مناسب توسط کامپایلر تولید می گردد. امتیاز استفاده از کلاس های کلی، کاهش حجم برنامه و قابلیت انعطاف زیاد می باشد.
- دستوراتی که در زیر آمده است، یک کلاس کلی با یک فیلد داده ای است که نوع آن در هنگام فرا خوانی مشخص می گردد:

```
template <class T >
class simple {
    private:
        T data;
    public:
        simple (T n);
        void show();
};
```

نوع واقعی بعداً تعیین می شود

هرنوع داده اولیه را می توان تعریف کرد

انواع داده انتزاعی کلی (ادامه)

- برای اعلان شیءای از این نوع، به صورت زیر عمل می شود:

```
نام کلاس <نوع > object;
```

- به عنوان مثال، دستور زیر شیء s را از نوع کلاس simple اعلان می کند که نوع فیلد داده ای آن int است:

```
Simple <int> s;
```

- اکنون با استفاده ار کلاس کلی، یک پشته کلی را ایجاد می کنیم:

```
template <class T>
class stack {
  private:
     T *x;
     int myTop;
     int max;
  public:
     stack();
     void push(T a);
     T top();
     void pop();
     int full();
     int empty();
};
```

انواع داده انتزاعی کلی (ادامه)

- توجه داشته باشید که توابع مربوط به کلاس های کلی باید از نوع کلی باشند. به این ترتیب، تابع سازنده stack و توابع push و top به صورت زیر تعریف می شوند (بقیه توابع نیز به همین صورت تعریف خواهند شد):

```
template <class T>
stack<T> :: stack
{
...
}
template <class T>
void stack<T>:: push(T a)
{
...
}
template <class T>
T stack<T> :: top()
{
...
}
```

انواع داده انتزاعی کلی (ادامه)

- اکنون با استفاده از اعلا ن های زیر می توان یک پشته از نوع int ، یک پشته از نوع float و یک پشته از نوع char ایجاد کرد:

```
stack <int> s1(10);
stack <float> s2(20);
stack <char> s3(30);
```

- پیاده سازی کلاس کلی به عنوان الگویی در زمان ترجمه عمل می کند. وقتی کامپایلر اعلان شیء ای از کلاس کلی را ببیند، ابتدا نوع را جایگزین می کند تا شیء کاملی به دست آید. از آن جا به بعد، مثل اشیای معمولی با آن ها برخورد می شود که شرح آن گذشت.

وراثت

- اطلاعات موجود در یک بخش از برنامه در بخشهای دیگر مورد استفاده قرار می گیرند.
- اغلب اطلاعات بطور ضمنی بین قطعات برنامه تبادل می شود.
- وراثت یعنی اخذ خواص و ویژگیهای یک قطع از برنامه توسط قطعه دیگر بر اساس رابطه ای که بین این قطعات وجود دارد.

```
{ int i, j;
    { int j, k;
      k = i + j; }
}
```

وراثت (ادامه)

- مشکل استفاده مجدد از نوع داده انتزاعی این است که، خواص و قابلیت های نوع موجود، کاملاً برای استفاده مجدد مناسب نیستند، بلکه معمولاً باید تغییراتی در آن ها ایجاد شود. این کار مستلزم وجود فردی است که با کد (نوع) موجود آشنایی داشته باشد. گاهی این تغییرات مستلزم اعمال تغییرات در برنامه های کاربردی است.

- مشکل دیگر برنامه نویسی با انواع داده انتزاعی این است که مستقل از یکد یگر تعریف می شوند و در یک سطح قرار دارند (سلسله مراتبی نیستند) . این کار، سازمان دهی برنامه را جهت تطبیق با مسئله مورد نظر دشوار می سازد. در بسیاری از موارد، مسئله ها دارای اشیایی هستند که با هم ارتباط دارند (خیلی شبیه به هم هستند)، به عنوان مثال، ممکن است مسئله به طور سلسله مراتبی با هم ارتباط داشته باشد.

وراثت (ادامه)

- مفهوم ***وراثت*** هر دو مشکل مربوط به استفاده مجدد از نوع داده انتزاعی را رفع می کند. وراثت موجب می شود تا یک نوع داده انتزاعی، داده ها و عملکرد نوع داده انتزاعی موجود را به ارث ببرد، به طوری که نیاز به انجام تغییرات درنوع موجود نباشد. با استفاده از مفهوم وراثت، می توان یک نوع داده انتزاعی موجود را برای رفع نیازهای مسئله جدید به کاربرد. علاوه بر این، وراثت چارچوبی را برای تعریف سلسله مراتب انواع داده انتزاعی فراهم می آورد.

- وراثت به طور کلی بر دو نوع است:
  - وراثت ***یگانه*** در این رابطه وراثت، کلاسی مثل x فقط وارث یک کلاس مثل y است.
  - وراثت ***چند گانه*** در این رابطه وراثت، کلاسی مثل x وارث چند کلاس است.

وراثت در C++

- برای این که کلاسی از کلاس دیگر مشتق شود، به صورت زیر عمل می شود:

```
class کلاس مشتق :کلاس پایه نوع دستیابی
{
...
}
```

- نوع دستیابی در این اعلان، یکی از واژه های public یا private است

```
class base {
    int x;
    int y;
     public:
        void get();
        void diaplay();
     protected:
        calculate();
};
class derived : public base
{
    int m, n;
    public:
       void color();
};
```

وراثت در C++ (ادامه)

- توجه داشته باشید که در C++ هم وراثت یگانه و هم وراثت چند گانه وجود دارد. در دستورات زیر، کلاس c3 از کلاس های c1 و c2 مشتق می شود:

```
class c1 {
...
}
class c2 {
...
}
class c3 : public c1, public c2
{
...
}
```

وراثت (ادامه)

کلاسهای مشتق (ادامه)

- پیاده سازی: در کلاس مشتق فقط اسامی ارثی از کلاس پایه به فضای نام محلی کلاس مشتق اضافه می شوند و اسمی عمومی برای کاربران آن کلاس قابل مشاهده اند.
- هر نمونه ای از کلاس حافظه داده مخصوص به خود را دارد که شامل داده ها و اشاره گرهایی به متدهای کلاس است.

وراثت (ادامه)

- به عنوان مثالی از پیاده سازی وراثت در C++، کلاس های زیر را ببینید:

```
class base
{
   int width;
   int length;
   public:
      void calculate();
   protected:
      int area;
};
class derived : public base
{
  int diameter;
  float perime;
public:
  void display();
};
```

وراثت (ادامه)

- در زمان ترجمه، کلاس base به صورت یک struct در می آید:

```
struct base
{
   int width;
   int high;
   int area;
};
```

- کلاس derived به یک ساختمان به صورت زیر تبدیل می شود:

```
struct derived {
   int width;
   int high;
   int area;
   int diameter;
   float perime;
};
```

وراثت (ادامه)

متدها

- وراثت متدها برای ایجاد اشیای جدید قدرت دیگری اعمال می کند که در بسته بندی موجود نیست.
- کلاس پایه می تواند متدهایی داشته باشد که همنام آن ها در کلاس های مشتق نیز موجود باشد.
- اینگونه متدها را در کلاس پایه متد مجازی می گوییم.

وراثت (ادامه)

انقیاد پویای متدها در C++

- دستورات زیر را در c++ ببینید:

```
class shape
{
  public:
     virtual void draw()
     {
       ...
     }
}
class circle : public shape
{
  public:
     void draw()
     {
       ...
     }
}
class square : public shape
{
  public:
     void draw()
     {
       ...
     }
}
```

وراثت (ادامه)

کلاسهای انتزاعی

- گاهی تعریف کلاسها می تواند به صورت یک قالب باشد به طوری که کلاسهای دیگری از آن ساخته شوند دو روش داریم:
  - ابر کلاسهای انتزاعی
  - وراثت mixin
    - مثال:

```
Deltaclass StackMod {
   int peek() { return storage[size].FromElem();}
}
class newqueue = class ElemStack + deltaclass StackMod
```

- امتیاز mixin این است که کلاس delta می تواند به هر کلاسی اعمال شود.

وراثت (ادامه)

اشیا و پیامها

- برنامه اسمالتاک مرکب از مجموعه ای از تعاریف کلاس است که حاوی اشیا داده و متدها است .
- در اسمالتاک دارای سه ویژگی:
  - تعریف کلاس
  - نمونه سازی اشیا
  - ارسال پیام
- در اسمالتاک سه نوع پیام داریم:
  - پیام یکانی
  - پیام دودویی
  - پیام کلمه کلیدی

وراثت (ادامه)

اشیا و پیامها

وراثت کلاس

- داده های اسمالتاک براساس سلسله مراتب کلاس مشخص می شوند.
- اگر هرمتدی که به شی ارسال می شود در آن کلاس تعریف نشده باشد به کلاس پدر ارسال می شود و این روند ادامه می یابد.

وراثت (ادامه)

مفاهیم انتزاع

چهار نوع رابطه وجود دارد:

- اختصاصی
- تجزیه
- نمونه سازی
- انفرادی سازی

چند ریختی

- استفاده از پارامترها در زیربرنامه ها قدیمی ترین ویژگی زبانهای برنامه سازی است
- چندریختی به توابعی اعمال می شود که یک نوع به عنوان آرگومان آنها است.
- زبانهای ام ال و اسمالتاک از چندریختی به بهترین شکل استفاده می کنند.

چند ریختی (ادامه)

پیاده سازی:

- زبانهایی که چند ریختی پویا را اجازه می دهند منجر به مشکل می شوند.

آرگومانها به دو شکل می توانند به تابع چند ریختی ارسال شوند:

- توصیفگر صریح
- توصیفگر فشرده

- آرگومانهای زیر می توانند به تابع چندریختی ارسال شوند:
  - داده صریح ۳۲ بیتی
  - داده کاراکتری ۸ بیتی
  - داده بولین یک بایتی
  - ساختار رکورد پیچیده

فصل ششم

کنترل ترتیب اجرا

کنترل ترتیب اجرا

دو جنبه کار :

- کنترل ترتیب اجرای عملیات که آن را *کنترل ترتیب* می نامیم.
- کنترل انتقال داده ها بین زیر برنامه ها و برنامه ها است که کنترل داده ها نامیده می شود.

کنترل ترتیب ضمنی و صریح

ساختارهای کنترل ترتیب به چهار دسته:

- کنترل ترتیب درسطح *عبارات* که با استفاده از تقدم عملگرها، پرانتزها و قواعد شرکت پذیری عملگرها انجام می گیرد.
- کنترل ترتیب در سطح *دستورات* که با ساختارهای شرطی، تکرار و انتخاب انجام می گیرد.
- *برنامه نویسی اعلانی* یک مدل اجرایی است که به دستورات بستگی ندارد اما موجب پیشروی کنترل اجرا در برنامه می شود.
- کنترل ترتیب در *زیربرنامه ها* که با استفاده از فراخوانی زیربرنامه ها و همروال ها که موجب انتقال کنترل از نقطه ای به نقطه دیگر می شود.

کنترل ترتیب ضمنی و صریح (ادامه)

ساختارهای کنترل ترتیب ممکن است ضمنی یا صریح باشد:

- ساختار کنترل ***ضمنی***
  - توسط زبان تعریف شده اند و بکار گرفته می شوند.
- ساختار کنترل ترتیب ***صریح***
  - برنامه نویس تهیه می کند تا ساختارهای ضمنی تعریف شده توسط زبان را عوض کند.

ترتیب اجرا در عبارات محاسباتی

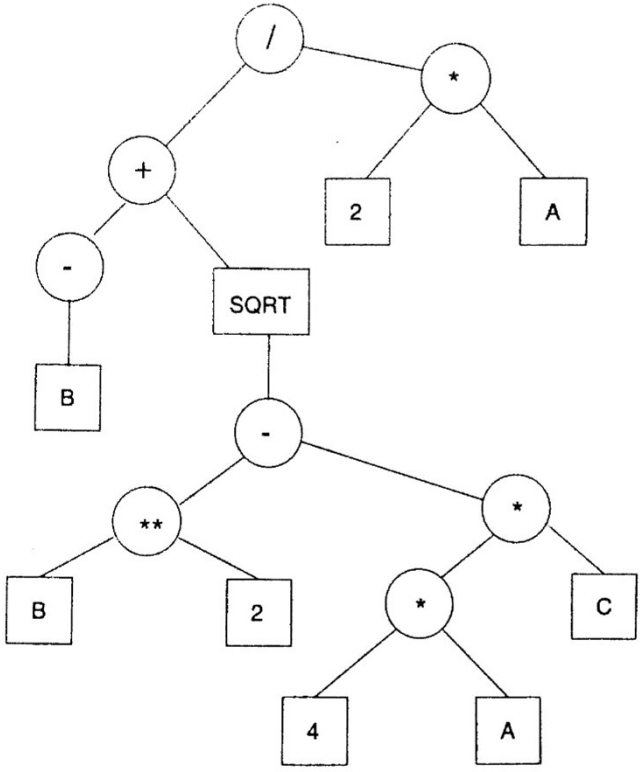
نمایش درختی عبارات

- با در نظر گرفتن عملیات در عبارات آرگومانهای عملیات را **عملوند** می نامیم.
  - عملوند ممکن است ثابت، اشیای داده، یا عملیات دیگری باشند.
- مکانیزم کنترل ترتیب در عبارات **ترکیب تابعی** است یعنی عملیات و عملوندهایش مشخص می شود.
- نمایش درختی ساختار کنترلی عبارت را نشان می دهد.

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

- نمایش درختی فرمول محاسبه ریشه معادله درجه دوم



ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

نحو عبارات

- اگر عبارات بصورت درختی نمایش داده شوند باید بتوان درختها را بصورت دنباله خطی از نمادها نشان داد.
- متداول ترین نشانه گذاریها عبارتند از:
  - نشانه گذاری prefix
  - نشانه گذاری Postfix
  - نشانه گذاری infix

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

معنای عبارات

- شکلهای نشانه گذاری ذکر شده در چگونگی ارزیابی عبارات با هم متفاوت هستند:
  - ارزیابی عبارات prefix
  - ارزیابی عبارات Postfix
  - ارزیابی عبارات infix

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

ارزیابی عبارات prefix

1. فراخوانی توابع بصورت نشانه گذاری prefix انجام می شود.
2. می تواند برای نمایش عملیات با هر تعداد از عملوندها به کار گرفته شود.
3. ترجمه عبارات prefix به دنباله ای از کد، ساده امکانپذیر است.

مورد سوم با استفاده از الگوریتم زیر انجام پذیر است. با فرض اینکه P یک عبارت prefix است:

1. اگر قلم بعدی در P عملگر است، آنرا در بالای پشته قرار دهید.
2. اگر قلم بعدی P یک عملوند باشد، آنرا در بالای پشته قرار دهید.
3. اگر n عنصر بالای پشته عملوندهایی باشند که مورد نیاز یک عملگر n تایی باشند، عملگر بالای پشته را می توانیم بر روی این عملوندها اعمال کنیم.

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

ارزیابی عبارات Postfix

1. اگر قلم بعدی Pیک عملوند است، آنرا در بالای پشته قرار دهید.

2. اگر قلم بعدی در P یک عملگر n تایی است، n آرگومان آن در بالای پشته قرار دارند. به جای این n قلم، نتیجه اجرای عملگر بر روی آنها را در بالای پشته قرار دهید.

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

ارزیابی عبارات infix

- استفاده از infix در برنامه سازی مشکلاتی را به وجود می آورد:
  1. نشانه گذاری infix برای عملگرهای دودویی مناسب است.
  2. وقتی بیش از یک عملگر infix در عبارتی ظاهر شود، این نشانه گذاری مبهم است، مگر اینکه از پارانتز استفاده شود.
- به همین دلیل قواعد ضمنی را تدارک می بینند تا نیازی به پارانتز نباشد
  - سلسله مراتب عملگرها (قواعد تقدم عملگرها)
  - شرکت پذیری

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

ارزیابی عبارات infix

- سلسله مراتب عملگرها (قواعد تقدم عملگرها)

$a + b \times c$

- فرض کنید $a = 3$، $b = 4$ و $c = 5$ باشد. اگر عملگرها از چپ به راست ارزیابی شوند نتیجه آن ۳۵ ولی اگر از راست به چپ ارزیابی شوند، نتیجه آن ۲۳ است.

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

ارزیابی عبارات infix

- شرکت پذیری
  - عبارات زیر را در نظر بگیرید:

$$a - b + c - d$$

- اگر سطح تقدم + و – یکسان باشد، قاعده تقدم نمی تواند ارزیابی این عبارت را مشخص کند.
- اگر دو عملگر با تقدم یکسان در عباراتی در کنار هم قرار گیرند، قاعده شرکت پذیری زبان مشخص می کند که کدام عملگر باید زودتر انجام شود.

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش زمان اجرا

- مترجم در اولین مرحله ساختار کنترلی درختی را برای عبارت ایجاد می کند.
  - اگر شامل نشانه گذاری infix باشد، از قواعد تقدم و شرکت پذیری ضمنی استفاده می کند.
- در مرحله اختیاری دوم، تصمیمات مشروح در رابطه با ترتیب ارزیابی گرفته می شود.

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش زمان اجرا (ادامه)

- به دلیل مشکل بودن رمزگشایی عبارت به شکل infix مطلوب است به شکل اجرایی تبدیل شود که در حین اجرا به راحتی رمزگشایی شود. گزینه های مختلف عبارتند از:
  - دنباله ای از کد ماشین
  - ساختارهای درختی
  - شکل Prefix or postfix

کنترل ترتیب بین دستورات

دستورات اصلی

- انتساب به اشیای داده
  - دستور انتساب
  - هدف اولیه انتساب مقدار راست عبارت را به مقدار چپ آن نسبت دهد.
  - دستورات ورودی
  - سایر عملیات انتساب
- شکلهای مختلف کنترل ترتیب سطح دستور
  - ترکیب (composition)
  - انتخاب (alternation)
  - تکرار (iteration)

کنترل ترتیب بین دستورات (ادامه)

دستورات اصلی (ادامه)

- کنترل ترتیب ضمنی
  - دستور goto
    - goto غیرشرطی
    - goto شرطی
  - دستور break

کنترل ترتیب بین دستورات (ادامه)

دستورات اصلی (ادامه)

طراحی برنامه نویسی ساخت یافته

- امتیازات goto
  - اگر برچسبها از نظر نحوی ساده باشندمستقیماً توسط سخت افزار پشتیبانی می شود و کارایی آن بالا است.
  - استفاده از آن در برنامه های کوچک ساده است.
  - برای برنامه نویسان اسمبلی و کسانی که با زبانهای قدیمی برنامه نویسی می کنند آشنا است.
  - هدف کلی برای نمایش شکلهای دیگری از کنترل است.

کنترل ترتیب بین دستورات (ادامه)

دستورات اصلی (ادامه)

طراحی برنامه نویسی ساخت یافته (ادامه)

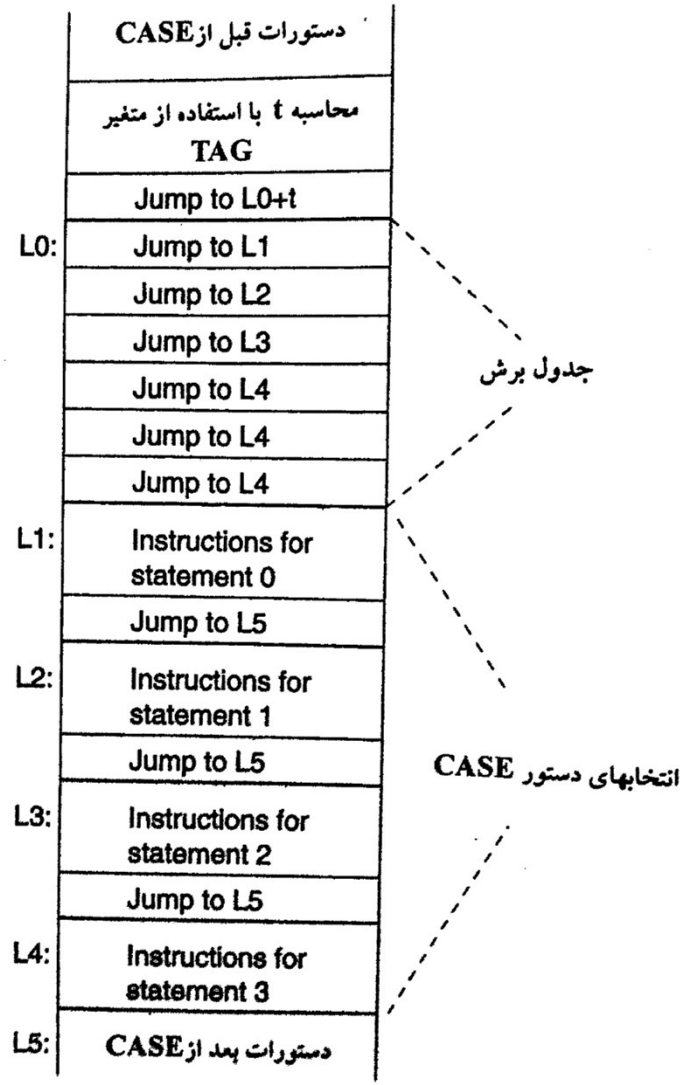
- معایب goto :
  - عدم وجود ساختار سلسله مراتبی برنامه
  - ترتیب دستورات در متن برنامه لازم نیست با ترتیب اجرا یکی باشد.
  - گروهی از دستورات ممکن است اهداف متعددی داشته باشد.

کنترل ترتیب بین دستورات(ادامه)

کنترل ترتیب ساخت یافته

- دستورات مرکب
  - دستور مرکب، دنباله ای از دستورات که به عنوان یک دستور محسوب می شود.
- دستورات شرطی
  - دستورات if
    - **if** condition **then** statement **endif**
    - **if** condition **then** statement$_1$ **else** statement$_2$ **endif**
  - دستورات case

کنترل ترتیب بین دستورات(ادامه)



کنترل ترتیب ساخت یافته (ادامه)

- پیاده سازی دستورات case
  - برای پیاده سازی از جدول پرش استفاده می شود.

شکل ۱- ۷ پیاده سازی جدول پرش دستور case

کنترل ترتیب بین دستورات(ادامه)

کنترل ترتیب ساخت یافته

- دستورات تکرار
  - تکرار ساده

perform body K **times**

  - تکرار در صورتی که شرط برقرار باشد.

while test **do** body

  - تکرار با افزایش یک شمارنده

for I:=1 **step** 2 **until** 30 **do** body
for K:=N-1 **step** 2×(W-1) **until** M×N **do** body

  - تکرار مبتنی بر دادهها

foreach $X (@arrayitem) {…}

  - تکرار نامتناهی

کنترل ترتیب بین دستورات(ادامه)

کنترل ترتیب ساخت یافته (ادامه)

- مشکلات کنترل ترتیب ساخت یافته
  - خروج چندگانه از حلقه

```
for I:=1 to K do
   if VECT[I] then goto a
```

  - Do-while-do

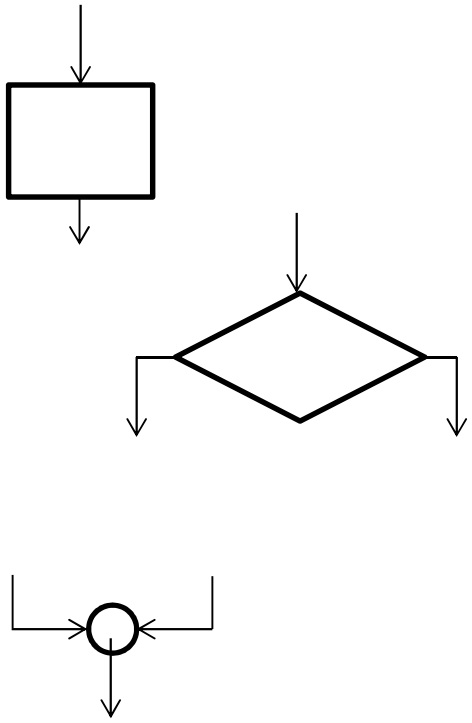
```
dowhiledo
   read(X)
   while( not end_of_file)
   process( X )
end dowhiledo
```

  - شرایط استثنایی

کنترل ترتیب بین دستورات (ادامه)

برنامه های بنیادی

- فرض می کنیم که گرافهای برنامه شامل سه دسته ازگره ها هستند.
  - گره های تابع
  - گره های تصمیم گیری
  - گره اتصال



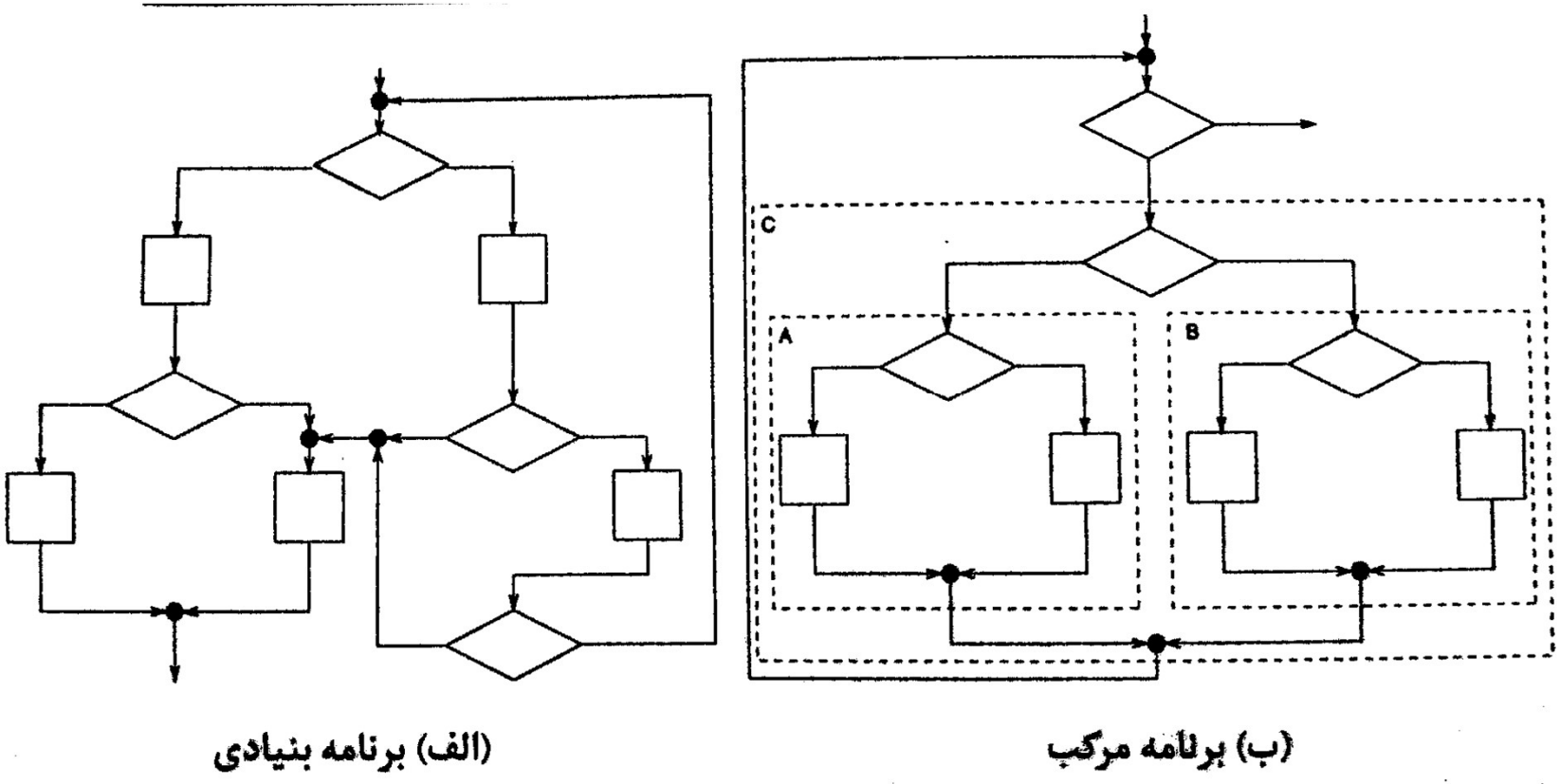
کنترل ترتیب بین دستورات (ادامه)

برنامه های بنیادی (ادامه)

- برنامه محض (proper)
  - برنامه ای بصورت فلوچارت است که مدل رسمی یک ساختار کنترلی است و:
    1. فقط یک کمان ورودی دارد.
    2. فقط یک کمان خروجی دارد.
    3. مسیری از کمان ورودی به هر کمان و از هر کمان به کمان خروجی وجود دارد.
  - برنامه بنیادی یک برنامه محض است که نمی تواند به برنامه های محض کوچکتر تقسیم شود.
  - برنامه مرکب یک برنامه محض است که بنیادی نیست.

کنترل ترتیب بین دستورات (ادامه)

برنامه های بنیادی (ادامه)



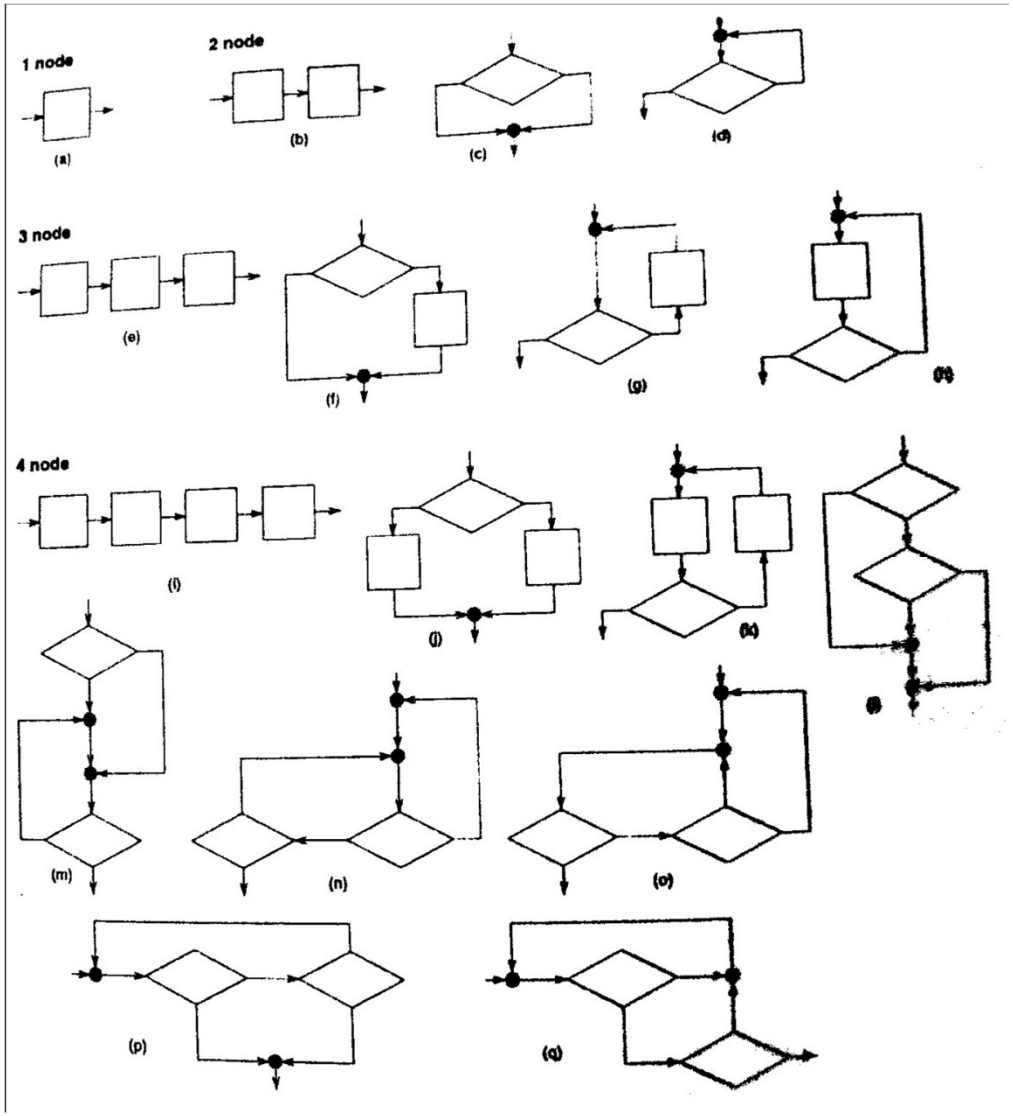
(ب) برنامه مرکب

(الف) برنامه بنیادی

شکل ۸-۸ فلوچارتها

کنترل ترتیب بین دستورات (ادامه)

برنامه های بنیادی (ادامه)



شکل ۹-۸ برنامه های بنیادی

فصل هفتم

کنترل زیربرنامه

کنترل ترتیب زیر برنامه

- زیربرنامه ساده فراخوانی – برگشت
  - هر برنامه متشکل از یک برنامه اصلی است که در حین اجرا می تواند زیربرنامه هایی را فراخوانی کند و هر زیربرنامه زیربرنامه های دیگر را .
- قاعده کپی
  - دستور فراخوانی زیربرنامه مثل این است که قبل از اجرا یک کپی از زیربرنامه در نقطه ای که فراخوانی صورت می گیرد قرار داده شود.

کنترل ترتیب زیر برنامه (ادامه)

فرضیه های موجود در دیدگاه کپی:

- زیربرنامه ها نمی توانند بازگشتی باشند.
- نیاز به دستور فراخوانی صریح است.
  - در حالی که پردازش استثناء نیازی به فرا خوانی صریح ندارد.
- زیربرنامه ها در هر فراخوانی باید به طور کامل اجرا شوند
  - اما زیر برنامه هایی که به عنوان همروال موزد استفاده قرار می گیرند چنین نیستند.

کنترل ترتیب زیر برنامه (ادامه)

فرضیه های موجود در دیدگاه کپی: (ادامه)

- کنترل به نقطه فراخوانی بر می گردد.
  - برای فراخوانی زیر برنامه زمانبندی شده، اجرای زیر برنامه ممکن است مدتی به تعویق افتد.
- در هر زمان فقط یک زیربرنامه کنترل را دردست دارد.
  - زیر برنامه هایی که به عنوان task مورد استفاده قرار می گیرند، ممکن است به طور همزمان اجرا شوند.

کنترل ترتیب زیر برنامه (ادامه)

زیربرنامه های فراخوانی - برگشت

- فرا خوانی زیر برنامه ها به دو شکل انجام می شود
  - فراخوانی تابعی
  - فراخوانی رویه یا زیر روال

کنترل ترتیب زیر برنامه (ادامه)

زیربرنامه های فراخوانی – برگشت (ادامه)

- پیاده سازی:
  - نیاز به چیزهای دیگر:
    - بین تعریف زیربرنامه و سابقه فعالیت آن تفاوت وجود دارد.
    - سابقه فعالیت دو بخش دارد :
      - سگمنت کد
      - رکورد فعالیت
    - سگمنت کد در حین اجرا تغیر نمی کند.
    - رکورد فعالیت در هر بار اجرای زیربرنامه ایجاد می شود و با خاتمه زیربرنامه از بین می رود.

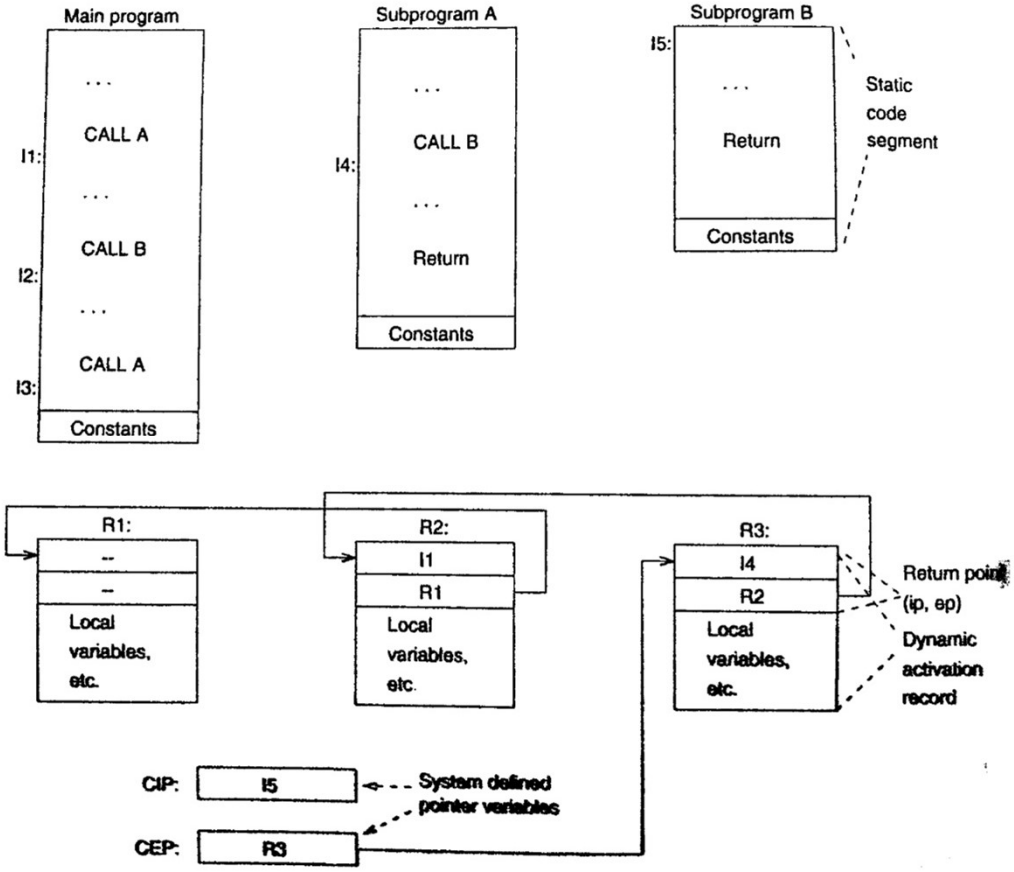
کنترل ترتیب زیر برنامه (ادامه)

زیربرنامه های فراخوانی – برگشت (ادامه)

- برای نگهداری نقطه ای که برنامه از آنجا اجرا می شود، به دو قلم اطلاعات نیاز داریم:
  - اشاره گر دستور فعلی CIP (current-instruction pointer)
  - اشاره گر محیط فعلی CEP (current environment pointer)

کنترل ترتیب زیر برنامه (ادامه)

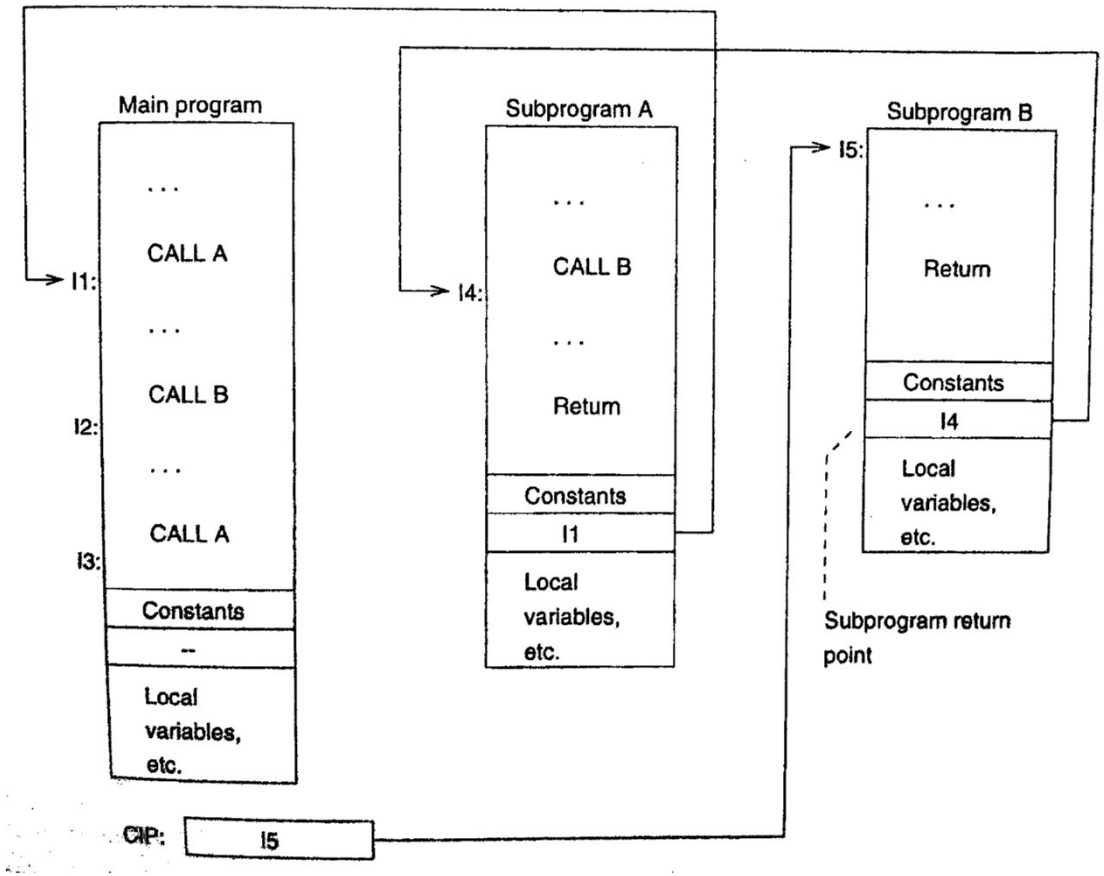
زیربرنامه های فراخوانی – برگشت (ادامه)



حالت اجرا در شروع اجرای زیر برنامه B

کنترل ترتیب زیر برنامه (ادامه)

زیربرنامه های فراخوانی – برگشت (ادامه)



کنترل ترتیب زیر برنامه (ادامه)

زیربرنامه های فراخوانی - برگشت (ادامه)

- پیاده سازی پشته ای
  - ساده ترین تکنیک مدیریت حافظه زمان اجرا پشته است.
  - برای کنترل مدیریت حافظه نیاز به اشاره گر پشته است.
  - در پاسکال یک پشته مرکزی و یک ناحیه حافظه به طور ایستا تخصیص می یابد.
  - پشته در لیسپ به صورت فراخوانیهای زیربرنامه به صورت تو در تو می باشد.

کنترل ترتیب زیر برنامه(ادامه)

زیربرنامه های بازگشتی

- مشخصات: اگر فراخوانی بازگشتی زیربرنامه امکانپذیر باشد A می تواند هر زیربرنامه ای از جمله خودش را فراخوانی کند.
- پیاده سازی: در هنگام فراخوانی هر زیربرنامه رکورد فعالیت جدیدی ایجاد می شود و با دستور برگشت از بین می رود.

کنترل ترتیب زیر برنامه(ادامه)

اعلان پیشرو در پاسکال

- اعلان پیشرو مثل امضای زیربرنامه است که شامل لیست پارامترها و کلمه forward است.

صفات کنترل داده ها

اسامی و محیطهای ارجاع

- اشیای داده به دو روش به عنوان عملوند یک عملیات مورد استفاده قرار می گیرند:
  - انتقال مستقیم
  - مراجعه از طریق شی داده ای که دارای نام است.
- انتقال مستقیم برای کنترل داده ها بین عابارت بکار می رود.

صفات کنترل داده ها (ادامه)

اسامی و محیطهای ارجاع (ادامه)

- عناصری از برنامه که دارای نام هستند (عناصر مشترک):
  - اسامی متغیرها
  - اسامی پارامترهای مجازی
  - اسامی زیربرنامه ها
  - اسامی انواع تعریف شده
  - اسامی ثوابت تعریف شده
  - برچسب دستورات
  - اسامی استثناها
  - اسامی عملیات اولیه مثل +و*وsort
  - اسامی ثوابت لیترال مثل ۲۵/۳و ۱۷

صفات کنترل داده ها (ادامه)

اسامی و محیطهای ارجاع (ادامه)

وابستگیها و محیطهای ارجاع

- کنترل داده ها به انقیاد شناسه ها به اشیای داده زیربرنامه ها مربوط می شود.
- این انقیاد را وابستگی می نامند و ممکن است به صورت جفتی از شناسه و شی داده یا زیربرنامه مربوط به آن نمایش داد.

صفات کنترل داده ها (ادامه)

اسامی و محیطهای ارجاع (ادامه)

وابستگیها و محیطهای ارجاع(ادامه)

- در حین اجرای برنامه در اغلب زبانها مشاهده می شود که:
  - در آغاز اجرای برنامه اصلی وابستگی شناسه ها ، نام هر متغیر تعریف شده در برنامه را ...
  - وقتی برنامه اصلی اجرا می شود عملیات ارجاعی را فراخوانی می کند ...
  - هر وقت زیربرنامه جدید فراخوانی می شود وابستگیهای دیگری برای ...
  - وقتی زیربرنامه اجرا می شود عملیات ارجاعی را فراخوانی می کند تا شی داده...
  - وقتی زیربرنامه کنترل را به برنامه اصلی برمی گرداند...
  - وقتی کنترل به برنامه اصلی برمیگردد ...

صفات کنترل داده ها (ادامه)

اسامی و محیطهای ارجاع (ادامه)

وابستگیها و محیطهای ارجاع(ادامه)

- مفاهیم اصلی کنترل داده:
- محیطهای ارجاع
  - محیط ارجاع محلی(یا محیط محلی)
  - محیط ارجاع غیرمحلی
  - محیط ارجاع عمومی
  - محیط ارجاع از پیش تعریف شده
- قابلیت مشاهده
- حوزه پویا
- عملیات ارجاع
- ارجاعهای محلی، غیرمحلی و عمومی

صفات کنترل داده ها (ادامه)

اسامی و محیطهای ارجاع (ادامه)

نام مستعار برای اشیای داده

- یک شی داده در طول عمرش ممکن است بیش از یک نام داشته باشد یعنی ممکن است چندین وابستگی در محیطهای ارجاع مشخص وجود داشته باشد.
- به دلیل مشکلاتی که نام مستعار ایجاد می کند طراحی زبان جدید سعی در حذف یا محدود کردن نام مستعار دارد.

صفات کنترل داده ها(ادامه)

حوزه ایستا و پویا

- حوزه پویای وابستگی مربوط به یک شناسه مجموعه ای از سابقه های فعالیت زیربرنامه است که وابستگی در حین اجرا قابل مشاهده است.
- قاعده حوزه پویا : حوزه پویای هر وابستگی را برحسب حالت پویای اجرای برنامه تعریف می کند.
- اهمیت حوزه ایستا: اغلب فرآیندها یکبار در زمان ترجمه انجام شوند.

صفات کنترل داده ها(ادامه)

ساختار بلوکی

- مفهوم ساختار بلوک در زبانهای ساخت یافته بلوگی مثل پاسکال پیدا شد.
- هر زیربرنامه یا برنامه به صورت مجموعه ای از بلوکهای تودرتو سازماندهی می شود.
- ویژگی مهم بلوک : محیط ارجاع جدیدی را معرفی میکند.
- با مجموعه ای از اعلان ها برای اسامی شروع می شود و سپس مجموعه ای از دستورات قرار می گیرد که به آن اسامی مراجعه می کنند.

صفات کنترل داده ها(ادامه)

داده های محلی و محیطهای ارجاع محلی

- محیط محلی زیربرنامه Q شامل شناسه های گوناگونی است که در عنوان زیربرنامه Q اعلان شده اند
- برای محیطهای محلی، قواعد حوزه پویا و ایستا سازگارند
- نگهداری: وابستگی X ممکن است نگهداری شود تا Q دوباره فراخوانی گردد
- حذف: وابستگی Xممکن است حذف شود.

صفات کنترل داده ها(ادامه)

داده های محلی و محیطهای ارجاع محلی(ادامه)

پیاده سازی: بهتر است محیط محلی زیربرنامه را به صورت جدول محیط ارجاع نشان داد.

- حافظه مربوط به هر شی به صورت یک نوع نمایش داده می شود و محل آن در حافظه به صورت مقدار چپ است .

نگهداری: اگر محیط ارجاع محلی زیربرنامه sub بین فراخوانیهای مختلف نگهداری شود فقط یک جدول محیط ارجاع محلی ایجاد می شود که حاوی متغیرهای نگهداری شده است.

صفات کنترل داده ها(ادامه)

داده های محلی و محیطهای ارجاع محلی(ادامه)

حذف: اگر محیط محلی sub در بین فراخوانیها حذف شود و هنگام ورود به آن دوباره ایجاد شودجدول محیط محلی حاوی متغیرهای حذف شده به عنوان بخشی از رکورد فعالیت sub تخصیص می یابد.

امتیازات و معایب: در نگهداری زیربرنامهایی که نوشته می شود نسبت به گذشته حساس است . و روش حذف موجب صرفه جویی در حافظه می شود

پارامترها و انتقال پارامترها

چهار روش اصلی برای محیطهای غیرمحلی مورد استفاده:

- محیطهای مشترک صریح و محیطهای غیرمحلی صریح
- حوزه پویا
- حوزه ایستا
- وراثت

پارامترها و انتقال پارامترها(ادامه)

پارامترهای مجازی و واقعی

- اصطلاحات آرگومان و نتیجه به داده هایی اطلاق می شود که با مکانیزمهای مختلفی به زیربرنامه ارسال و از آن دریافت می شود.
- پارامترهای مجازی نوعی شی داده محلی در یک زیربرنامه است.
- پارامترهای واقعی یک شی داده است که با زیربرنامه فراخوان مشترک است.

پارامترها و انتقال پارامترها (ادامه)

پارامترهای مجازی و واقعی (ادامه)

- اصطلاحات آرگومان و نتیجه به داده هایی اطلاق می شود که با مکانیزمهای مختلفی به زیربرنامه ارسال و از آن دریافت می شود.
- پارامترهای مجازی نوعی شی داده محلی در یک زیربرنامه است.
- پارامترهای واقعی یک شی داده است که با زیربرنامه فراخوان مشترک است.

پارامترها و انتقال پارامترها (ادامه)

پارامترهای مجازی و واقعی (ادامه)

تناظر بین پارامترهای مجازی و واقعی

- تناظر موقعیتی
- تناظر براساس نام

پارامترها و انتقال پارامترها(ادامه)

روشهای انتقال پارامترها

توضیح فرآیند دو مرحله ای شامل:

- توصیف پیاده سازی جزئیات مکانیزم انتقال پارامتر
- توصیف معنای چگونگی استفاده از پارامترها
- چهارروش متداول :
  - فراخوانی با نام
  - فراخوانی با ارجاع
  - فراخوانی با مقدار
    - فراخوانی با مقدار و نتیجه
    - فراخوانی با مقدار ثابت
  - فراخوانی با نتیجه

پارامترها و انتقال پارامترها(ادامه)

انتقال معنا

- انواع داده اولیه با پارامتر in با فراخوانی مقدار ثبات و با پارامتر out یا in-out با فراخوانی مقدار و نتیجه ارسال می شوند.
- انواع داده مرکب به فراخوانی ارجاع ارسال می شوند.

مقادیر تابع

- مقادیر برگشتی به عنوان مقدار تابع هستند یعنی از طریق پارامتر برگردانده نمی شوند.

پارامترها و انتقال پارامترها(ادامه)

پیاده سازی انتقال پارامتر

- چون هر سابقه فعالیت زیربرنامه مجموعه متفاوتی از پارامترها را دریافت می کند حافظه پارامترهای مجازی زیربرنامه به عنوان بخشی از رکورد فعالیت زیربرنامه تخصیص می یابد هر پارامتر مجازی یک شی داده محلی در زیربرنامه است.

پارامترها و انتقال پارامترها(ادامه)

پیاده سازی انتقال پارامتر

مثالهایی از انتقال پارامترها

- متغیرهای ساده و ثوابت
- ساختمان داده ها
- عناصر ساختمان داده ها
- عناصر آرایه با اندیسهای محاسبه شده
- اشاره گرها
- نتایج عبارات
- نام مستعار و پارامترها
  - پارامتر مجازی و متغیر غیرمحلی
  - دو پارامتر مجازی

پارامترها و انتقال پارامترها(ادامه)

پیاده سازی انتقال پارامتر(ادامه)

زیربرنامه ها به عنوان پارامتر

- دو مشکل عمده با پارامترهای زیربرنامه :
  - کنترل نوع ایستا
  - ارجاعهای غیرمحلی

برچسب دستورات به عنوان پارامتر

- کدام سابقه فعالیت باید مورد استفاده قرار گیرد؟
- چگونه Goto به یک برچسب پیاده سازی می شود؟

محیطهای مشترک صریح

- مشخصات: محیط مشترک معادل محیطی برای یک زیربرنامه است با این تفاوت که بخشی از یک زیربرنامه خاص نیست.
- پیاده سازی: در فرترن و C هر زیربرنامه ای که از محیط مشترک استفاده می کند اعلانهایی برای متغیرهای مشترک دارد .

محیطهای مشترک صریح (ادامه)

اشتراک صریح متغیرها

- به جای اینکه گروهی از متغیرها در محیط مشترک و جدا از زیربرنامه ها باشند هر متغیر دارای یک مالک است و آن زیربرنامه است که در آنجا اعلان می شود.
- پیاده سازی: اثر اشتراک صریح متغیر مشابه استفاده از یک متغیر در محیط مشترک است .

محیطهای مشترک صریح

حوزه پویا

- قاعده تازه ترین وابستگی: در زنجیره پویایی از فراخوانی زیربرنامه ها که ازP شروع شد از تازه ترین وابستگی ایجاد شده برای X استفاده می کنیم .
- پیاده سازی: پیاده سازی آن با توجه به پیاده سازی پشته مرکزی برای ذخیره رکوردهای فعالیت زیربرنامه ساده است.

محیطهای مشترک صریح(ادامه)

حوزه ایستا و ساختار بلوکی

- محیط ارجاع غیرمحلی هر زیربرنامه در حین اجرا با استفاده از قواعد حوزه پویا تعیین می شود که در زمان ترجمه صورت می گیرد.
- ترتیب جدولهای محلی در پشته تودر تویی پویای سابقه های فعالیت زیربرنامه را نمایش می دهد.
- برای پیاده سازی کامل لازم است ساختار بلوک ایستا در حین اجرا طوری نمایش داده شود که بتواند ارجاع غیرمحلی را کنترل کند.

محیطهای مشترک صریح(ادامه)

حوزه ایستا و ساختار بلوکی (ادامه)

پیاده سازی زنجیر ایستا

- اشاره پر زنجیر ایستا همیشه حاوی آدرس پایه جدول محلی دیگری است که در محل پایینتر جدول قرار دارد.
- اشاره گرهای زنجیر ایستا مبنایی برای الگوی ارجاع است.

محیطهای مشترک صریح(ادامه)

حوزه ایستا و ساختار بلوکی (ادامه)

پیاده سازی: برای بهبود پیاده سازی به نکاتی نیاز داریم:

- هر زیربرنامه ای مثل R که اجرا می شود طول زنجیر پویا که جدول محلی R به طرف پایین پشته شروع می شود ثابت است ...
- در این زنجیر با طول ثابت یک ارجاع غیرمحلی همواره در یک نقطه از زنجیر برآورده می شود.
- موقعیتی از زنجیر ایستا که ارجاع محلی در آنجا برآورده خواهد شد می تواند در زمان ترجمه مشخص شود.

محیطهای مشترک صریح(ادامه)

حوزه ایستا و ساختار بلوکی (ادامه)

وابستگی مناسب در دو مرحله انجام می شود:

- مقدار ورودی اول را به عنوان اندیش نماشگر در نظر بگیرید
- محل ورودی مطلوب را با استفاده از آدرس پایه و آفست به دست آورید.

محیطهای مشترک صریح(ادامه)

حوزه ایستا و ساختار بلوکی (ادامه)

اعلانها در بلوکهای محلی

باتوجه به ماهیت پویای فراخوانی زیربرنامه ها نمی توانیم مطمئن باشیم که کدام زیربرنامه ها فعلاً در حال اجرا است.

پایان