بسمه تعالی

دانشگاه آزاد اسلامی کردکوی

طراحی و پیاده سازی زبانهای برنامه سازی

بر اساس کتاب:

اصول طراحی و پیاده سازی زبانهای برنامه سازی

ترجمه جعفرنژاد قمی

مدرس: محمد جلالی

فصل اول

اصول طراحی زبانها

۱–۱– 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.

۱–۲– تاریخچه مختصری از زبانهای برنامه سازی

a) توسعه زبانهای اولیه

- زبانهای مبتنی بر اعداد (اواخر دهه ۱۹۳۰ تا اوایل دهه ۱۹۴۰)
- اهداف الگول عبارت بودند از:
  - نشانه های الگول باید به ریاضیات استاندارد نزدیک باشد.
  - الگول باید برای توصیف الگوریتمها مفید باشد.
  - برنامه ها در الگول باید به زبان ماشین ترجمه شوند.
  - الگول نباید به معماری یک ماشین مقید باشد.

۱–۲– تاریخچه مختصری از زبانهای برنامه سازی

a) توسعه زبانهای اولیه (ادامه)

- زبانهای تجاری (۱۹۵۵)
- زبان هوش مصنوعی (دهه ۱۹۵۰)
- زبانهای سیستم

۱–۲– تاریخچه مختصری از زبانهای برنامه سازی (ادامه)

b) تکامل معماری نرم افزار

دوران کامپیوترهای بزرگ

- محیط دسته ای
- محیط محاوره ای
- تاثیر بر طراحی زبان

دوران کامپیوتر شخصی

- کامپیوترهای شخصی
- محیطهای سیستم تعبیه شده
- تاثیر بر طراحی زبان

۱–۲– تاریخچه مختصری از زبانهای برنامه سازی (ادامه)

b) تکامل معماری نرم افزار(ادامه)

دوران شبکه بندی

- محاسبات توزیعی
- اینترنت
- تاثیر بر زبان برنامه سازی

۱–۲– تاریخچه مختصری از زبانهای برنامه سازی (ادامه)

C) دامنه های کاربرد

کاربردها در دهه ۱۹۶۰

- پردازش تجاری
- محاسبات علمی
- برنامه نویسی سیستم
- کاربردهای هوش مصنوعی

۱–۲– تاریخچه مختصری از زبانهای برنامه سازی (ادامه)

C) دامنه های کاربرد(ادامه)

کاربردهای قرن ۲۱

- پردازش تجاری
- محاسبات علمی
- برنامه نویسی سیستم
- کاربردهای هوش مصنوعی
- انتشارات
- فرآیند : اغلب از یک برنامه برای کنترل برنامه ی دیگر استفاده می شود. مانند پاسخ خودکار به میل ها
- کاربردهای جدید (مانند شی گراهاو...):مانند کاربرد ام ال در تحقیقات زبانهای برنامه سازی برای بررسی تئوری نوع

۱–۳– نقش زبانهای برنامه سازی

تغییرات بوجود آمده و اثرات آنها بر زبانهای برنامه سازی

- تغییر در قابلیتهای کامپیوتر(کامپیوترهای بزرگ ، کند و گرانقیمت که از لامپ خلا استفاده می کردند به ریز کامپیوترها و سوپر کامپیوترها تبدیل شدند) : ساختار و هزینه های استفاده از زبانهای سطح بالا تحت تاثیر قرار گرفت.
- زمینه های کاربرد جدید: موجب طراحی زبانهای جدید ، ارتقاء و بازبینی زبانهای قدیمی
- یافتن متدهای برنامه نویسی خوب برای برنامه های بزرگ و پیچیده و تغییر در محیط های برنامه نویسی: موجب رشد در طراحی زبان ها شد.
- متدهای پیاده سازی : انتخاب ویژگیهای نو برای طراحی های جدید
- مطالعات تئوری: استفاده از متدهای رسمی ریاضیات
- نیاز به انتقال برنامه از کامپیوتری به کامپیوتر دیگر: موجب استانداردسازی در زبا نها

۱–۳– نقش زبانهای برنامه سازی(ادامه)

a) زبان خوب چگونه است ؟

صفات یک زبان خوب

- وضوح ، سادگی و یکپارچگی :

جامعیت مفهومی : مفاهیم و ابزارهای موجود در یک زبان و قوانین ترکیب آنها در یک زبان برنامه سازی

خوانایی برنامه : تفاوتهای معنایی منعکس کننده تفاوتهای نحوی باشد.

- قابلیت تعامد : امکان ترکیب ویژگیهای مختلف زبان و با معنا بودن ترکیب حاصل

مثال : ترکیب عبارت وساختار 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.

۱–۳– نقش زبانهای برنامه سازی(ادامه)

صفات یک زبان خوب(ادامه)

- پشتیبانی از انتزاع
- سهولت در بازرسی برنامه
- محیط برنامه نویسی : وجود ویراستارهای خاص ،امکانات نگهداری و اصلاح نسخه های متفاوت
- قابلیت حمل برنامه
- هزینه استفاده
  - **هزینه اجرای برنامه : بستگی به کامپایلر دارد ولی امروزه زیاد مهم نیست.**
  - **هزینه ترجمه برنامه: در برنامه های دانشجویی برنامه به تعداد زیاد ترجمه میشود تا اجرا**
  - **هزینه نگهداری برنامه:هزینه های ترمیم خطا بعد از اجرا ،توسعه و تغییر سیستم عامل و . .**

۱–۳– نقش زبانهای برنامه سازی(ادامه)

نحو 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.

۱–۳– نقش زبانهای برنامه سازی(ادامه)

b) مدلهای زبان

- زبانهای دستوری(imperative) یا رویه ای: زبانهای مبتنی بر فرمان یا دستورگرا
  مانند C , C++ و پاسکال و . . .
- زبانهای تابعی(applicative): به جای مشاهده تغییر حالت عملکرد برنامه دنبال می شود.
  مانند ام ال و لیسپ (بعضی وقتها C) $function_n(...(function_2(function_1(data)) ...)$
- زبانهای قانونمند(rule-based): شرایطی را بررسی می کنند و درصورت برقرار بودن آنها فعالیتی را انجام می دهند.
  - مانند پرولوگ $enable\ condition_1$ $action_1$
- 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.

۱–۳– نقش زبانهای برنامه سازی(ادامه)

C) استاندارد سازی زبان

روش پی بردن به معنای دستورات :

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

استانداردهای زبان دو دسته اند :

- استاندارد خصوصی : توسط شرکت یا مالک زبان ارائه می شوند.
- استاندارد عمومی : اسنادی که توسط سازمانهای مختلف به توافق رسیده اند.

مسائل مهم در استفاده ی موثر از استاندارد:

- زمان سنجی : چه زمانی باید زبان استاندارد شود ؟
- اطاعت و پیروی : 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 ؟

۱–۳– نقش زبانهای برنامه سازی(ادامه)

d) بین المللی شدن برنامه نویسی

- ترتیب تلفیق: کاراکترها به چه ترتیبی باید ظاهر شوند؟
- ترتیب: موقعیت کاراکترهای غیر رومی
- حالت کاراکترها: حروف کوچک و بزرگ در زبانهایی مثل ژاپنی ، عربی و یهودی
- جهت پیمایش: اغلب زبانها از چپ به راست خوانده می شوند.
- فرمت تاریخ در یک کشور خاص
- فرمت زمان در یک کشور خاص
- مناطق زمانی
- سیستمهای حروفی
- علامت پول

۱-۴- محیط های برنامه نویسی

a) محیط برنامه نویسی در دو زمینه بر طراحی زبان تاثیر گذاشته است :

- کامپایل کردن مجزای زیربرنامه و سایر بخشهای برنامه
  - کامپایلر باید این اطلاعات را داشته باشد:

    مشخه ی تعداد ، ترتیب و نوع پارامترهای زیربرنامه

    اعلان نوع داده

    تعریف نوع داده
- تست و اشکال زدایی
  - مانند : ویژگیهای ردیابی اجرا ، نقاط کنترلی ، ادعا

۱-۴- محیط های برنامه نویسی(ادامه)

b) محیط های کاری

- محیط کاری ، خدماتی مثل ذخیره داده ها ، رابط گرافیکی کاربر ، امنیت و خدمات ارتباطی را فراهم می کند.

۱-۴- محیط های برنامه نویسی(ادامه)

C) زبانهای کنترل کار و فرآیند

- مفهوم کنترل کار به چارچوبهای محیط برمی گردد.
- کاربر کنترل مستقیم بر روی مراحل مختلف برنامه دارد.

فصل دوم

اثرات معماری ماشین

مقدمه

- در توسعه ی یک زبان برنامه نویسی سه عامل بر روی طراحی زبان موثر است :
  - کامپیوتری که برنامه بر روی آن اجرا می شود
  - مدل اجرا یا کامپیوتر مجازی که آن زبان را بر روی سخت افزار اجرا می کند.
  - مدل محاسباتی که زبان آن را پیاده سازی می کند

مقدمه

- کامپیوتر ها می توانند در یکی از سه شکل زیر باشند :
  - کامپیوتر واقعی (یا سخت افزاری)
  - کامپیوتر شبیه سازی شده ی نرم افزاری (میان افزار)
  - کامپیوتر مجازی که ترکیبی از سخت افزار و نرم افزار است
  - ترکیبی از روشهای بالا

عملکرد کامپیوتر

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

- هر کامپیوتر از ۶ جزء تشکیل شده است:
  - داده ها : داده ها و ساختمان داده ها
  - اعمال اولیه
  - کنترل ترتیب
  - دستیابی به داده ها
  - مدیریت حافظه
  - محیط عملیاتی : مکانیزمی برای ارتباط با محیط خارجی که حاوی داده و برنامه است.

عملکرد کامپیوتر(ادامه)

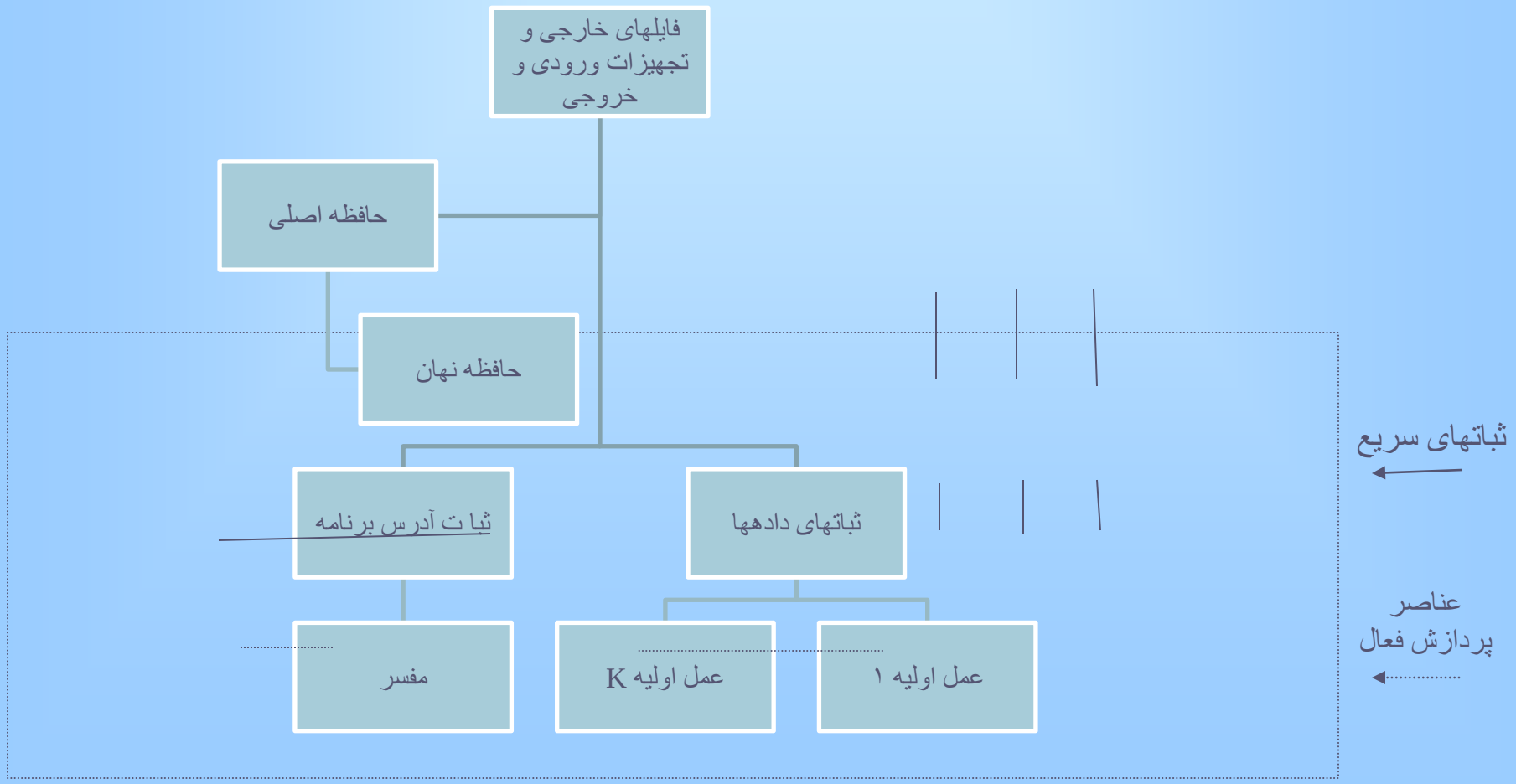
سخت افزار کامپیوتر

۱) داده ها

- سه جزء اصلی حافظه داده ها :
  - حافظه اصلی ، ثباتهای سریع و فایلهای خارجی

- انواع داده در کامپیوتر :
  - نوع داده توکار : مانند انواع داده اولیه
  - نمایش زبان ماشین کامپیوتر : نمایش توکار برای برنامه

سازمان یک کامپیوتر معمولی



عملکرد کامپیوتر(ادامه)

سخت افزار کامپیوتر (ادامه)

۲) اعمال اولیه : کامپیوتر باید مجموعه ای از اعمال اولیه توکار داشته باشد که متناظر با کدهای عملیاتی هستند که به صورت دستورات زبان ماشین می باشند.

اعمال اولیه برای انجام محاسبات - اعمال اولیه برای تست خواصی از داده های اولیه- اعمال اولیه برای دستکاری قسمتی از عناصر داده ها - اعمال اولیه برای کنترل دستگاه های جانبی-اعمال اولیه برای کنترل ترتیب اجرا

۳) کنترل ترتیب: در حین اجرای برنامه دستور بعدی که باید اجرا شود توسط محتویات ثبات آدرس برنامه مشخص می گردد. این ثبات حاوی آدرس دستور بعدی است.

عملکرد کامپیوتر(ادامه)

سخت افزار کامپیوتر (ادامه)

۴) دستیابی به داده ها : علاوه بر کد عملیاتی هر دستور ماشین باید عملوندهایی را مشخص کند که آن عمل از آن استفاده می کند. عملوند ممکن است در حافظه اصلی یا در ثبات باشد.

مکانیزمی که برای تعیین عملوند وبازیابی آن و ذخیره نتایج انجام می گیرد کنترل دستیابی به داده ها گویند راه حل استفاده از آدرس در حافظه و ثبات است.

۵) مدیریت حافظه: تمام منابع کامپیوتر (مثل حافظه ، پردازنده مرکزی ، دستگاههای حافظه خارجی) تا آنجایی که ممکن است فعال باشند. عدم توازن سرعت بین پردازنده و حافظه اصلی و داده خارجی

- **برای برقرای توازن بین پردازنده و داده خارجی از چند برنامگی و برای چند برنامگی از صفحه بندی استفاده می شود.**
- **برای برقرای توازن بین پردازنده و حافظه اصلی از حافظه نهان استفاده می شود.**

عملکرد کامپیوتر(ادامه)

سخت افزار کامپیوتر (ادامه)

۶) محیط عملیاتی : متشکل از مجموعه ای از حافظه جانبی و دستگاههای ورودی و خروجی است. این دستگاه ها محیط خارج از کامپیوتر را نشان می دهند و هر ارتباطی با کامپیوتر از طریق محیط عملیاتی صورت می گیرد. مثل حافظه های سریع ، حافظه هایی با سرعت متوسط ، حافظه های کند و دستگاههای ورودی و خروجی

عملکرد کامپیوتر (ادامه)

کامپیوترهای میان افزار

- کامپیوتر میان افزار ، توسط ریز برنامه ای 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.
- کامپیوتری را که از طریق شبیه سازی ریزبرنامه ای بوجود می آید کامپیوتر مجازی گویند چون از طریق ریزبرنامه ی شبیه سازی شده بوجود آمده است و بدون این ریزبرنامه ، ماشین وجود نخواهدداشت.

عملکرد کامپیوتر (ادامه)

معماریهای مجازی

دو روش برای اجرای برنامه سطح بالا در کامپیوتر مجازی :

۱) ترجمه (کامپایل کردن)

۲) شبیه سازی نرم افزاری (تفسیر نرم افزاری)

عملکرد کامپیوتر (ادامه)

۱) 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 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

عملکرد کامپیوتر (ادامه)

معماریهای مجازی

- انواع مفسر :
  - **اسمبلر : مفسری است که زبان منبع آن اسمبلی و زبان مقصد آن زبان ماشین است**
  - **کامپایلر : مفسری است که زبان منبع آن سطح بالا و زبان مقصد آن نزدیک به زبان ماشین است**
  - **بارکننده یا ویراستار پیوند: مفسری است که زبان منبع آن کد ماشین و زبان مقصد آن مشابه ورودی است**
  - **پیش پردازنده یا پردازنده ماکرو : مفسری است که زبان منبع آن شکل توسعه یافته ای از سطح بالا و زبان مقصد آن شکل استاندارد آن زبان سطح بالا می باشد.**

عملکرد کامپیوتر (ادامه)

مفسرها و معماریهای مجازی (ادامه)

۲) 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.

- با اضافه کردن زیربرنامه هایی به زبان ماشین در کامپیوتر میزبان کاری می کنیم تا الگوریتم های مورد نیاز برای اجرای زبان سطح بالا بوجود آیند

عملکرد کامپیوتر (ادامه)

مفسرها و معماریهای مجازی (ادامه)

- زبانها به دو دسته هستند:
  - زبان های کامپایلری : C,C++ ، فرترن ، پاسکال و ادا . برنامه های آن قبل از شروع اجرای برنامه به زبان ماشین کامپیوتر واقعی ترجمه می شوند به طوریکه شبیه سازی به مجموعه ای از روالهای پشتیبانی زمان اجرا محدود می شود که اعمال اولیه موجود در زبان منبع را شبیه سازی می کند که شباهت زیادی به زبان ماشین ندارد.
    - **اجرای سریعتر**
  - زبان های مفسری: لیسپ ، ام ال ، پرل ، پست اسکریپت ، پرولوپ و اسمالتاک معمولاً با مفسر نرم افزاری پیاده سازی می شود.مترجم کد ماشین را برای کامپیوتر تولید نمی کند مفسر شکل میانی از برنامه را تولید می کند.
    - **اجرای کندتر ،این زبانها مترجم های ساده تری دارند**

کامپیوترهای مجازی و زمانهای انقیاد

روشهای ساخت کامپیوتر:

- از طریق سخت افزار : ساختمان داده ها والگوریتم ها مستقیما با دستگاههای سخت افزاری نمایش داده می شوند.
- از طریق میان افزار: ساختمان داده ها والگوریتم ها از طریق ریز برنامه نویسی نمایش داده می شوند.
- از طریق ماشین مجازی: ساختمان داده ها والگوریتم ها از طریق برنامه نویسی در زبانهای دیگر نمایش داده می شوند.
- از طریق ترکیبی از این تکنیکها : بخشهای مختلف کامپیوتر مستقیما در سخت افزار یا به وسیله شبیه سازی نرم افزاری نمایش داده می شوند.

کامپیوترهای مجازی و زمانهای انقیاد (ادامه)

کامپیوترهای مجازی و پیاده سازی های زبان

- سه عامل منجر به 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 ، منجر به تفاوتهایی می شود .

کامپیوترهای مجازی و زمانهای انقیاد (ادامه)

کامپیوترهای مجازی و پیاده سازی های زبان

- بعنوان مثال اگر کامپیوتر مجازی ، حاوی عمل جمع صحیح و عمل جذرگیری باشد ، پیاده ساز ممکن است جمع صحیح را به وسیله سخت افزار و جذرگیری را به وسیله نرم افزار انجام دهد.

کامپیوترهای مجازی و زمانهای انقیاد

انقیاد : محدود کردن یک عنصر برنامه به ویژگی یا صفت خاص را گویند.

زمان های انقیاد:

- زمان اجرا
  - در ورود به زیر برنامه یا بلوک:انقیاد پارامترهای مجازی به واقعی در C
  - در نقطه خاصی از اجرای برنامه:انقیاد متغییر به مقدار
- زمان ترجمه (زمان کامپایل)
  - انقیاد توسط برنامه نویس انتخاب می شود: اسامی متغییرها و نوع متغییر ها
  - انقیاد توسط مترجم انجام می شود:محل نسبی شی داده
  - انقیادهایی که توسط بارکننده صورت می گیرد.مترجم در حافظه ای که به زیربرنامه اختصاص می یابد متغییرها را به آدرس آنها مقید میکند.
- زمان پیاده سازی زبان: جزئیات مربوط به نمایش داده ها و اعمال محاسباتی
- زمان تعریف زبان:اغلب ساختارهای زبان برنامه نویسی در زمان تعریف زبان انجام می گیرد مثل شکلهای مختلف دستورات ، انواع ساختمان داده ،ساختارهای برنامه

کامپیوترهای مجازی و زمانهای انقیاد (ادامه)

انقیاد و زمان انقیاد (ادامه)

x:=x + 10

- در برنامه ای که به زبان L نوشته می شود انقیادها و زمانهای انقیاد عناصر زیر بحث می شود:
  - 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 و زمان انقیاد (ادامه)

x:=x + 10

- در برنامه ای که به زبان L نوشته می شود انقیادها و زمانهای انقیاد عناصر زیر بحث می شود:
  - مجموعه ای از انواع ممکن برای متغیر X : در زمان تعریف زبان
  - نوع متغیر : در زمان ترجمه زبان
  - مجموعه ای از مقادیر ممکن برای X : در زمان پیاده سازی زبان
  - مقدار متغیر X : در زمان اجرای برنامه
  - نمایش مقدار ثابت ۱۰ : در زمان تعریف زبان
  - خواص عملگر +: در زمان تعریف زبان

کامپیوترهای مجازی و زمانهای انقیاد (ادامه)

اهمیت زمانهای امقیاد

- انقیاد زودرس: زبانهایی که در آنها انقیاد در زمان ترجمه انجام می شود
  - **مثال :فرترن و پاسکال و C**
  - **مزایا : کارایی بالاتر (برای مثال کار با آرایه های بزرگ و محاسبات راحتر)**
- انقیاد دیررس: زبانهایی که در آنها انقیاد در زمان اجرا انجام می شود
  - **مثال : ام ال و لیسپ**
  - **مزایا : انعطاف بیشتر (برای مثال دستکاری رشته راحتر)**
- زبانی مثل ادا که هم کارایی و هم انعطاف مهم است می توان زمان انقیاد را انتخاب کرد.

فصل سوم

اصول ترجمه زبان

فهرست مطالب

- نحو زبان برنامه نویسی
  - معیار عمومی نحو
  - عناصر نحوی زبان
  - ساختار برنامه .زیربرنامه
- مراحل ترجمه ی برنامه
  - تحلیل برنامه منبع
  - ترکیب برنامه مقصد

نحو زبان برنامه سازی

- نحو آرایش واژه ها به عنوان عناصری از یک دنباله است ، که آرایش واژه ها رابطه بین آنها را نشان می دهد.
- دستور x=y+z در c معتبراست ولی x=yz+- معتبر نیست.

- برای توصیف یک زبان برنامه سازی به بیش از نحو یک زبان نیاز داریم.
- نحو دستور x=4.34 + 5.86 مشخص نمی کند که . . .

نحو زبان برنامه سازی(ادامه)

معیار عمومی نحو

- قابلیت خوانایی
- قابلیت نوشتن
- سهولت بازرسی
- سهولت ترجمه
- عدم وجود ابهام

نحو زبان برنامه سازی(ادامه)

معیار عمومی نحو

- قابلیت خوانایی

اگر ساختار مربوط به الگوریتم و دادهای استفاده شده در برنامه به خوبی روشن باشد خوانایی بالاست.

نحو زبان برنامه سازی(ادامه)

معیار عمومی نحو

- قابلیت خوانایی
- قابلیت نوشتن

در تضاد با قابلیت نوشتن است

مثال : تعریف متغییر در فرترن

نحو زبان برنامه سازی(ادامه)

معیار عمومی نحو

- قابلیت خوانایی
- قابلیت نوشتن
- سهولت بازرسی

با قابلیت خوانایی و نوشتن در ارتباط است

نحو زبان برنامه سازی(ادامه)

معیار عمومی نحو

- قابلیت خوانایی
- قابلیت نوشتن
- سهولت بازرسی
- سهولت ترجمه

خوانایی و نوشتن مربوط به برنامه نویس و سهولت ترجمه مربوط به مترجم است

نحو زبان برنامه سازی(ادامه)

معیار عمومی نحو

- قابلیت خوانایی
- قابلیت نوشتن
- سهولت بازرسی
- سهولت ترجمه
- عدم وجود ابهام

مثال ifهای تودرتو و حل آن در پاسکال ادا و الگول

نحو زبان برنامه سازی (ادامه)

عناصر نحوی زبان

- کاراکترها
- شناسه ها
- نمادهای عملگر
- کلمات کلیدی و کلمات رزروی
- کلمات اضافی

کلمات اختیاری که برای افزایش خوانایی است

مثال : go to

نحو زبان برنامه سازی (ادامه)

عناصر نحوی زبان (ادامه)

- توضیحات
- فضای خالی
- فاصله ها و محصور کننده ها
- فرمتهای آزاد و طول ثابت

فرمت آزاد : یعنی دستورات برنامه می توانند از هر جایی از خط شروع شود

- عبارت
- دستورات

نحو زبان برنامه سازی (ادامه)

عناصر نحوی زبان (ادامه)

- توضیحات
- فضای خالی
- فاصله ها و محصور کننده ها
- فرمتهای آزاد و طول ثابت
- عبارت

عملیات اصلی برای تغییر حالت ماشین هستند.

- دستورات

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه
- تعریف داده ها به صورت جداگانه
- تعریف زیربرنامه به صورت تودر تو
- تعریف واسط مجزا
- توصیف داده ها جدا از دستورات اجرایی است
- تعریف زیربرنامه ها به طور غیرمجزا

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه

پیوند زیربرنامه ها در هنگام بارکردن

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه
- تعریف داده ها به صورت جداگانه

داده های مربوط به یک شی جدا هستند مثل کلاس در C++

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه
- تعریف داده ها به صورت جداگانه
- تعریف زیربرنامه به صورت تودر تو

در دوران اولیه ی زبانها ،زیربرنامه ها دارای محیط ارجاع غیرمحلی بودند مثل : فرترن ،الگول

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه
- تعریف داده ها به صورت جداگانه
- تعریف زیربرنامه به صورت تودر تو
- تعریف واسط مجزا

مانند **header** فایل ها در **C**

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه
- تعریف داده ها به صورت جداگانه
- تعریف زیربرنامه به صورت تودر تو
- تعریف واسط مجزا
- توصیف داده ها جدا از دستورات اجرایی است

برای مثال در کوبول برنامه سه قسمت دارد بخش داده ها ،بخش رویه ها ، بخش داده های خارجی

نحو زبان برنامه سازی (ادامه)

ساختار برنامه - زیربرنامه

- تعریف زیربرنامه ها به صورت جداگانه
- تعریف داده ها به صورت جداگانه
- تعریف زیربرنامه به صورت تودر تو
- تعریف واسط مجزا
- توصیف داده ها جدا از دستورات اجرایی است
- تعریف زیربرنامه ها به طور غیرمجزا

تمایز خاصی بین دستورات برنامه اصلی و زیربرنامه ها وجود ندارد مثل اسنوبال ۴

فهرست مطالب

- نحو زبان برنامه نویسی
  - معیار عمومی نحو
  - عناصر نحوی زبان
  - ساختار برنامه .زیربرنامه
- مراحل ترجمه ی برنامه
  - تحلیل برنامه منبع
  - ترکیب برنامه مقصد

۳-۲- مراحل ترجمه

در زبانی که به صورت مفسری پیاده سازی شود سرعت اجرای برنامه پایین خواهد بود.

- فرآیند ترجمه به طور منطقی به دو مرحله :
  - تحلیل برنامه منبع ورودی
  - ترکیب برنامه مقصد اجرایی

مراحل ترجمه (ادامه)

- کامپایلر استاندارد دوگذره:
  - گذر تحلیل: برنامه را به اجزا تشکیل دهنده آن تجزیه می کند
  - گذر دوم: با استفاده از این اطلاعات جمع آوری شده برنامه مقصد را تولید می کند.

مراحل ترجمه (ادامه)

تحلیل برنامه منبع

- تحلیل لغوی : دسته بندی از کاراکترها به اجزای بنیادی
- تحلیل نحوی (تجزیه) : ساختارهای بزرگ با استفاده از عناصر لغوی که توسط تحلیل گر لغوی تولید شدند شناسایی می شوند.
- تحلیل معنایی : ساختارهای معنایی که توسط تحلیلگر نحوی تشخیص داده شدند پردازش می شوند و ساختار کد مقصد اجرایی شکل می گیرد.

مراحل ترجمه (ادامه)

تحلیل برنامه منبع (ادامه)

- متداولترین اعمال در زمان تحلیل برنامه :
  - نگهداری جدول نماد
  - درج اطلاعات ضمنی
  - کشف خطا
  - پردازش ماکرو و عملیات زمان ترجمه

مراحل ترجمه (ادامه)

ترکیب برنامه مقصد

- بهینه سازی
- تولید کد
- پیوند زدن و بار کردن

فصل چهارم

انواع داده اولیه

فهرست مطالب

- خواص انواع داده :

اشیاء داده

مقادیر ثابت و متغییرها

انواع داده و اعلان

کنترل نوع و تبدیل نوع

انتساب و مقداردهی

- انواع داده اسکالر :

انواع داده عددی

انواع داده شمارشی

انواع داده بولی

انواع داده کارکتری

- انواع داده مرکب :

رشته های کارکتری

اشاره گرها و اشیاء داده برنامه نویس

فایلهای ورودی و خروجی

مقدمه

- 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

- از اصطلاح شی داده برای گروهبندی زمان اجرای یک یا چند قطعه از داده ها در کامپیوتر مجازی استفاده می کنیم.
- بعضی اشیا در حین اجرای برنامه توسط برنامه نویس تعریف شده اند. مثل متغییر
  - این اشیا توسط برنامه نویس تغییر می یابند.
- بعضی از اشیای داده توسط سیستم تعریف می شوند. مثل پشته ی زمان اجرا
  - اجزای تعریف شده توسط سیستم در حین اجرای برنامه در صورت نیاز به طور خودکار ایجاد می شوند.

خواص انواع و اشیاء (ادامه)

اشیای داده

- شی داده ظرفی برای مقادیر داده است یعنی محلی که مقادیر در آن ذخیره می شود. شی داده توسط مجموعه ای از صفات بیان می شود که مهمترین آن نوع داده می باشد.
- مقدار داده ممکن است یک عدد ، کاراکتر یا اشاره گری به شی داده دیگر باشد.
- اگر شی داده حاوی مقداری باشد که همیشه به عنوان یک واحد دستکاری شود آن را شی داده اولیه گویند.
- اگر شی داده مجموعه ای از سایر اشیای داده باشد ساختمان نامیده می شود.

خواص انواع و اشیاء (ادامه)

اشیای داده

- شی داده در طول عمر خود انقیادهای گوناگونی می پذیرد که مهمترین آنها عبارتند از:
  - نوع
  - محل
  - مقدار
  - نام
  - اجزاء

فهرست مطالب

- خواص انواع داده :

اشیاء داده
مقادیر ثابت و متغییرها
انواع داده و اعلان
کنترل نوع و تبدیل نوع
انتساب و مقداردهی

- انواع داده اسکالر :

انواع داده عددی
انواع داده شمارشی
انواع داده بولی
انواع داده کارکتری

- انواع داده مرکب :

رشته های کارکتری
اشاره گرها و اشیاء داده برنامه نویس
فایلهای ورودی و خروجی

خواص انواع و اشیاء (ادامه)

متغیرها و ثوابت

- شی داده ای که توسط برنامه نویس تعریف و نامگذاری می شود متغیر نام دارد.
- ثابت :یک شی داده ی با نام است که مقداری به آن نسبت داده می شود و در طول عمر آن ثابت است
  - ثابت لیترال: ثابتی است که نامش همان نمایش مقدارش است
  - ثابت تعریف شده توسط برنامه نویس: ثابتی است که نامش در تعریف شی داده توسط برنامه نویس انتخاب می شود.
- انقیاد ثابت به مقدار توسط مترجم صورت می گیرد.

خواص انواع و اشیاء (ادامه)

متغیرها و ثوابت (ادامه)

- **ماندگاری :** داده ها ماندگارند ولی متغییرها عمر محدود دارند.
- اغلب برنامه های امروزی هنوز براساس مدل پردازش دسته ای نوشته می شوند.

یعنی برنامه نویس دنباله از رویدادهای زیر را فرض می کند:

- برنامه به حافظه بار می شود.
- داده خارجی مناسب(دیسک و نوار) برای برنامه مهیایند.
- داده ورودی موردنظر خوانده شده در متغیرهایی در حافظه قرار می گیرند. متغییرها دستکاری و نتیجه بر می گردد.
- برنامه خاتمه می یابد.

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

خواص انواع و اشیاء (ادامه)

انواع داده

- نوع داده طبقه ای از اشیای داده به همراه مجموعه ای از عملیات 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.

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

- عناصر اصلی مشخصات یک نوع داده:
  - صفاتی
  - مقادیری
  - عملیاتی

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

- عناصر اصلی پیاده سازی یک نوع داده:
  - نمایش حافظه ای برای ذخیره سازی
  - شیوه ای که عملیات تعریف شده برای نوع داده را تعریف می کند.

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

مشخصات انواع داده اولیه:

- صفات
- مقادیر
- عملیات

خواص انواع و اشیاء (ادامه)

انواع داده (ادامه)

چهارعامل موجب می شوند تا تعریف عملیات زبان برنامه سازی دشوار شود:

- عملیاتی که برای ورودیهای خاصی تعریف نشده اند.
- آرگومانهای ضمنی
- اثرات جانبی (نتایج ضمنی):مانند تغییر ورودیهای تابع
- خود اصلاحی (حساسیت به سابقه)

اگر نوعی به عنوان بخشی از نوع بزرگتر باشد آن را **زیر نوع** و نوع بزرگتر را **ابرنوع** می گویند.

خواص انواع و اشیاء (ادامه)

پیاده سازی انواع داده اولیه

شامل نمایش حافظه مربوط به اشیاداده ای ،مقادیر نوع داده ای و عملیاتهای دستکاری نوع داده است.

- نمایش حافظه: حافظه مربوط به انواع داده اولیه تحت تاثیر کامپیوتری است که برنامه را اجرا می کند.

صفات اشیای داده اولیه با روشهای زیر پیاده می شوند:

- برای کارایی بعضی از زبانها طوری طراحی شدند که صفات داده ها توسط کامپایلر تعیین شوند. مثل Cو پاسکال و فرترن
- صفات شی داده ممکن است در زمان اجرا دریک توصیف گر و به عنوان بخشی از شی داده ذخیره شود. برای انعطاف پذیری . مثل لیسپ و پرولوگ

خواص انواع و اشیاء (ادامه)

پیاده سازی انواع داده اولیه (ادامه)

- پیاده سازی عملیات: هر عملیاتی که برای نوعی از اشیای داده تعریف می شود.
- 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

- دستوری از برنامه است که نام و نوع اشیای داده را که در حین اجرای برنامه مورد نیاز هستند مشخص می کند.
- انواع اعلان داده: صریح ، ضمنی
- اشیایی که در طول عمرشان به اسامی مانند A,B مقید می شوند اعلان صریح گویند.مانند :
  int a;
- در بعضی از زبانها اعلان ضمنی یا اعلان پیش فرض وجود دارد. مانند متغییر INDEX در فرترن

خواص انواع و اشیاء (ادامه)

اعلانها

اعلان عملیات

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

- اهداف اعلان:
  - انتخاب نمایش حافظه
  - مدیریت حافظه
  - عملیات چندریختی
  - کنترل نوع

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع

- کنترل نوع : یعنی هر عملیاتی که در برنامه انجام می گیرد تعداد و نوع آرگومانهای آن درست باشد.
- کنترل نوع ممکن است در زمان اجرا صورت گیرد(کنترل نوع پویا)
- کنترل نوع ممکن است در زمان ترجمه صورت گیرد(کنترل نوع ایستا)

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

معایب کنترل نوع پویا:

- اشکالزدایی 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 بالا
- کنترل نوع پویا باید به صورت نرم افزاری پیاده سازی می شود.سرعت پایین

مزایای کنترل نوع پویا:

- قابلیت انعطاف در طراحی برنامه

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

کنترل نوع ایستا:

- اطلاعات مورد نیاز در خصوص نوع ،از اعلان برنامه نویس یا از ساختار زبان بدست می آید.

بدست آوردن اطلاعات مورد نیاز برای کنترل نوع ایستا:

- برای هر عمل تعداد و ترتیب و نوع آرگومانها و نتایج
- برای هر متغییر نام نوع شی داده
- نوع هر شی داده ثابت

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

مزایای کنترل نوع ایستا:

- مصرف حافظه پایین
- سرعت بالا

معایب کنترل نوع ایستا:

- انعطاف پایین در طراحی برنامه
- بدلیل وجود برخی از ساختارهای زبان ،در بعضی از موارد کنترل نوع ایستا امکان ندارد

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

برای برطرف کردن معایب کنترل نوع ایستا دو روش:

- کنترل نوع پویا
- عملیات کنترل نشوند.

خواص انواع و اشیاء (ادامه)

کنترل نوع و تبدیل نوع (ادامه)

تبدیل نوع و تبدیل نوع ضمنی

اگر در حین کنترل نوع ،نوع مورد انتظار و نوع واقعی یکسان نباشند موارد زیر اتفاق می افتد :

- 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 مجموعه 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.

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

خواص انواع و اشیاء (ادامه)

انتساب و مقدار دهی اولیه

- انتساب عملیات اصلی برای تغییر انقیاد یک مقدار به یک شی داده است.
- این تغییر اثر جنبی عملیات است.

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 شی داده مقدار چپ محاسبه شده نسبت بده
- مقدار راست محاسبه شده را به عنوان نتیجه عملیات برگردان

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده اسکالر

- داده های اسکالر از معماری سخت افزار کامپیوتر پیروی می کنند.
- داده های مرکب معمولا ساختار پیچیده ای دارند که توسط کامپایلر پیاده سازی می شوند و توسط سخت افزار پیاده نمی شوند

انواع داده اسکالر

- انواع داده اسکالر :
  - انواع داده عددی

اعداد صحیح

اعداد حقیقی ممیز شناور

اعداد حقیقی ممیز ثابت

اعداد موهومی

اعداد گویا

  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری

انواع داده اسکالر(ادامه)

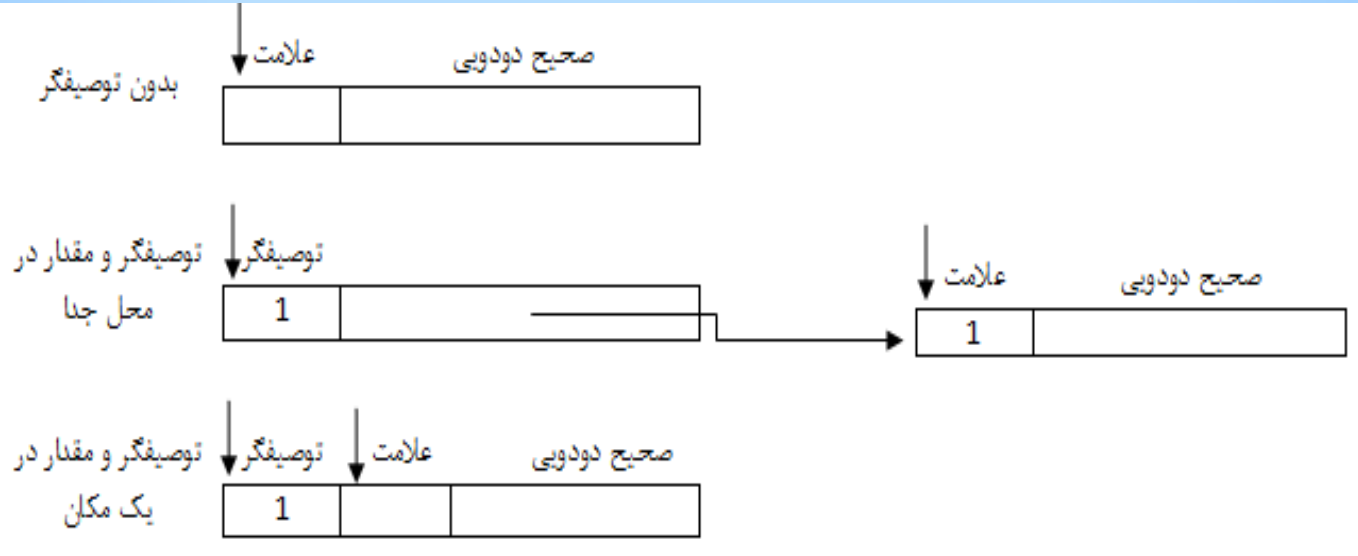
انواع صحیح :

- مشخصات : مهمترین صفت برای یک شی داده از نوع صحیح ، نوع است.
- عملیات بر روی اشیای داده صحیح شامل موارد زیر است:
  - عملیات محاسباتی
  - عملیات رابطه ای
  - انتساب
  - عملیات بیتی

انواع داده اسکالر(ادامه)

انواع صحیح :

- پیاده سازی : بعد از تعریف در زبان ، نمایش حافظه و عملیات بر روی آنها بصورت سخت افزاری پیاده می شود.



سه نمایش حافظه برای اعداد صحیح

انواع داده اسکالر(ادامه)

انواع صحیح :

زیر بازه ها:

- 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.
- انواع زیربازه دو اثر مهم در پیاده سازی دارد:
  - نیاز به حافظه کمتر
  - کنترل نوع بهتر

انواع داده اسکالر

- انواع داده اسکالر :
  - انواع داده عددی

اعداد صحیح
اعداد حقیقی ممیز شناور
اعداد حقیقی ممیز ثابت
اعداد موهومی
اعداد گویا

  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری

انواع داده اسکالر(ادامه)

اعداد حقیقی ممیز شناور

- 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.

انواع داده اسکالر

- انواع داده اسکالر :
  - انواع داده عددی

اعداد صحیح

اعداد حقیقی ممیز شناور

اعداد حقیقی ممیز ثابت

اعداد موهومی

اعداد گویا

  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری

انواع داده اسکالر(ادامه)

اعداد حقیقی ممیز ثابت

- مشخصات : اغلب سخت افزارها شامل اشیا داده صحیح و ممیز شناور هستند.ولی در بعضی از کاربردها نیاز داریم از اعداد حقیقیی با ممیز ثابت استفاده کنیم. مثلا برای نمایش دلار وسنت
  - تعریف عدد با ممیز ثابت در کوبول

```
X  picture 999V99
```

متغییر X عددی حقیقی با ممیز ثابت تعریف می شود که سه رقم قبل و دو رقم بعد از ممیز دارد

- پیاده سازی: ممکن است مستقیماً توسط سخت افزار پشتیبانی شود یا به صورت نرم افزاری شبیه سازی گردد.

انواع داده اسکالر

- انواع داده اسکالر :
  - انواع داده عددی

اعداد صحیح

اعداد حقیقی ممیز شناور

اعداد حقیقی ممیز ثابت

اعداد موهومی

اعداد گویا

  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری

انواع داده اسکالر(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.

  پیاده سازی : بصورت نرم افزاری

  نمایش حافظه : بصورت دو بلوک حافظه جدا

- اعداد گویا: عدد گویا خارج قسمت دو مقدار صحیح است.

  پیاده سازی : بصورت نرم افزاری

  نمایش حافظه : بصورت لیست پیوندی

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده اسکالر(ادامه)

نوع شمارشی

- 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 مانند زیر در C مشخص می کند.

```
enum emp {female, male};
```

- پیاده سازی: نمایش حافظه برای شی داده ای از نوع شمارشی ساده است. هر مقدار با اعداد صحیح نمایش داده می شود.

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده اسکالر(ادامه)

نوع بولی

- 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 از واحد حافظه برای این مقادیر استفاده می شود.
  - مقدار صفر در کل واحد حافظه نشاندهنده false و مقدار غیرصفر نشاندهنده true است.

بعضی از زبانها مانند C نوع بولی ندارند پاسکال و ادا نوع بولی را با انواع شمارشی پیاده می کنند

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده اسکالر(ادامه)

کاراکترها

- 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.

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده مرکب

۱) رشته های کاراکتری

شی داده ای است که از دنباله ای از کارکترها تشکیل شده است

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
- 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

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده مرکب(ادامه)

۲) اشاره گرها و اشیای داده برنامه نویس

برای اینکه زبان نوع اشاره گر را داشته باشد باید ویژگیهای زیر را دارا 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. (کنترل نوع ایستا)
- اشاره گرها ممکن است به هر نوع شی داده مراجعه کنند.(کنترل نوع پویا)

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 واقعی
- آدرس نسبی : آفستی از آدرس بلوک پایه

فهرست مطالب

- خواص انواع داده :
  - اشیاء داده
  - مقادیر ثابت و متغییرها
  - انواع داده و اعلان
  - کنترل نوع و تبدیل نوع
  - انتساب و مقداردهی
- انواع داده اسکالر :
  - انواع داده عددی
  - انواع داده شمارشی
  - انواع داده بولی
  - انواع داده کارکتری
- انواع داده مرکب :
  - رشته های کارکتری
  - اشاره گرها و اشیاء داده برنامه نویس
  - فایلهای ورودی و خروجی

انواع داده مرکب(ادامه)

۳) 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)

- متداول ترین فایلها ، فایلهای ترتیبی اند.
- فایلهای متنی
- ورودی – خروجی محاوره ای
- فایلهای دستیابی مستقیم
- فایلهای ترتیبی شاخص دار

انواع داده مرکب(ادامه)

فایلها و ورودی – خروجی (ادامه)

فایلهای ترتیبی

- ساختمان داده ای مرکب از دنباله خطی از عناصر همنوع است.
- طول آن متغیر است و حد بالایی ندارد.
- برای ورودی–خروجی داده ها معمولاً به صورت کاراکتری اند.
- فایل می تواند در حالت خواندن یا در حالت نوشتن دستیابی شود.

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی (ادامه)

فایلهای ترتیبی (ادامه)

- مشخصات: عملیات اصلی بر روی فایلهای ترتیبی :
  - بازکردن
  - خواندن
  - نوشتن
  - تست انتهای فایل
  - بستن

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی (ادامه)

فایلهای ترتیبی (ادامه)

- پیاده سازی: سیستم عامل مسئول پیاده سازی فایلها است.

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی (ادامه)

فایلهای متنی

- فایلی از کاراکترها است .
- شکل اولیه فایل مربوط به ورودی – خروجی در اغلب زبانها است.
- فایلهای متنی را مستقیماً از طریق صفحه کلید می توان ایجاد و چاپ کرد.
- این فایلها شکلی از انواع فایلهای ترتیبی هستند که یک سری عملیات ویژه را پشتیبانی می کنند

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی (ادامه)

ورودی – خروجی محاوره ای

- اصلاح چندین جنبه از دیدگاه معمولی فایلهای ترتیبی که در بالا مطرح شدند :
- فایل همزمان باید در حالت خواندن و نوشتن باشد.
- بافر کردن داده در ورودی و خروجی محدود می شود.
- اشاره گر موقعیت فایل و تست انتهای فایل ارزش چندانی ندارند.

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی (ادامه)

فایلهای دستیابی مستقیم

- در فایل ترتیبی عناصر به ترتیبی که در فایل قرار دارند بازیابی می شوند .
- دستیابی تصادفی به عناصر غیر ممکن است.
- می توان به هر عنصر به طور تصادفی دست یافت.
- به صورت مجموعه ای از عناصر نامرتب سازماندهی می شود.

انواع داده مرکب(ادامه)

فایلها و ورودی - خروجی (ادامه)

فایل ترتیبی شاخص دار

- این سازمان فایل مصالحه ای را بین سازمانهای ترتیبی محض و دستیابی مستقیم محض به وجود می آورد.
- این نوع فایل امکان دستیابی ترتیبی به عناصر بعدی را فراهم می کند که عنصر فعلی بطور تصادفی انتخاب شده است.

فصل ششم

بسته بندی

مقدمه

تمام فعالیتهای طراحی را می توان به عنوان طراحی مشخصات نوع داده انتزاعی در نظر گرفت یعنی:

- طراحی صفات
- عملیات موردنیاز

چهار روش ایجاد انواع داده جدید و عملیات بر روی آنها:

- ساختمان داده
- زیربرنامه ها
- اعلان نوع
- وراثت

ساختمان داده ها

اشیای داده ساختاری و انواع داده

- شی داده ای که مرکب از اشیای داده دیگری است ساختمان داده نام دارد.
- بسیاری از مفاهیم و اصول مربوط به ساختمان داده ها در زبانهای برنامه سازی مشابه اشیای داده اولیه است

ساختمان داده ها (ادامه)

مشخصات انواع ساختمان داده

صفات اصلی مشخص کننده ساختمان داده:

- تعداد عناصر
- نوع هر عنصر
- اسامی برای انتخاب عناصر
- حداکثر تعداد عناصر
- سازمان عناصر

ساختمان داده ها (ادامه)

مشخصات انواع ساختمان داده (ادامه)

عملیات در ساختمان داده ها

بعضی از عملیاتها در ساختمان داده ها از اهمیت ویژه ای برخوردارند:

- عملیات انتخاب عناصر
- عملیات بر روی کل ساختمان
- درج و حذف عناصر
- ایجاد و حذف ساختمان داده ها

ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها

نمایش های حافظه

شامل:

- حافظه ای برای عناصر ساختمان داده
- توصیفگر اختیاری آنها

دو نمایش اصلی:

- نمایش ترتیبی
- نمایش پیوندی

ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها (ادامه)

دو موضوع که انتخاب نمایش حافظه را تحت تاثیر قرار می دهد:

- انتخاب کارآمد عنصر از ساختمان
- مدیرحافظه کارآمد برای پیاده سازی زبان

ساختمان داده ها (ادامه)

پیاده سازی انواع ساختمان داده ها (ادامه)

نکات مهم در 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.

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)

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 موردنظر 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.

به علت تاثیر متقابل بین طول عمر شی داده و مسیرهای دستیابی دو مسئله مهم در مدیریت حافظه به وجود می آید:

- زباله
- ارجاعهای معلق

ساختمان داده ها (ادامه)

اعلانها و کنترل نوع برای ساختمان داده ها

- اعلان مثل انواع داده اولیه است ولی ساختمان داده ها پیچیده ترند زیرا صفات بیشتری باید مشخص شوند.
- کنترل نوع مثل انواع داده اولیه است ولی ساختمان داده ها پیچیده ترند 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.

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

مشخصات بردارها:

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

عملیات بر روی بردارها:

- عملیاتی که عنصری را از برداری انتخاب می کند اندیس گذاری نام دارد.
- برای ذخیره صفات بردار می توان از توصیفگر استفاده کرد.
  - **عملیات بر روی کل بردار**

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

پیاده سازی بردارها:

- دستیابی به عناصر
- 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
- **پیاده سازی 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 .

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

برش آرایه

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

ساختمان داده ها (ادامه)

بردارها و آرایه ها (ادامه)

آرایه های شرکت پذیر

- از طریق نام بتوان به اطلاعات دست یافت.

```
%CLIST ={"ali",'a',"bahram",'b'};
```

- از نام بعنوان اندیش استفاده شود.
- 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

- 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 دارای نام هستند.
- پیاده سازی: نمایش حافظه برای رکورد شامل یک بلوک از حافظه است که عناصر درآن به ترتیب ذخیره می شود.

ساختمان داده ها (ادامه)

رکوردها

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)

رکوردها

رکوردهای طول متغیر

- در رکوردهای طول متغیر عناصر ممکن است در یک زمان وجود داشته باشند و در زمان دیگر وجود نداشته باشند.برای حل مشکل:
  - کنترل پویا
  - کنترلی انجام نشود.

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.

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 نیازدارد:
  - یک فیلد نوع
  - دو فیلد اشاره گر لیست

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

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**

ساختمان داده ها (ادامه)

مجموعه ها(ادامه)

- تکنیکهای مقابله با برخورد:
  - درهم سازی مجدد
  - پیمایش ترتیبی
  - باکت بندی

ساختمان داده ها (ادامه)

اشیای داده اجرایی

- در اغلب زبانها ، برنامه های اجرایی و اشیای داده ای که توسط آنها دستکاری می شوند ساختارهای مجزایی هستنداما همیشه اینطور نیست.
- در پرولوگ عملیات consult وجود دارد.

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 کالماً 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 آنهابه 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
- طراحی ماژول معمولاً به دوروش انجام می شود:
  - ماژولهای تجزیه تابعی
  - ماژولهای تجزیه داده ای

انواع داده انتزاعی (ادامه)

پنهان سازی اطلاعات (ادامه)

- فلوچارت انتزاعی از ساختار کنترل سطح دستور برنامه است.
- روشهای طراحی برنامه ها:
  - اصلاح مرحله ای
  - برنامه نویسی ساخت یافته
  - برنامه نویسی پیمانه ای
  - برنامه نویسی بالا به پایین

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 آن می کند.
- زیربرنامه ها مکانیزم بسته بندی را شکل می دهند که تقریباً در هر زبانی وجود دارد.

بسته بندی با زیربرنامه ها

- دو دیدگاه از زیربرنامه در اینجا مهم است:
  - سطح طراحی برنامه
  - سطح طراحی زبان

بسته بندی با زیربرنامه ها(ادامه)

زیر برنامه ها و عملیات انتزاعی

- مشخصات زیربرنامه:
  - نام
  - امضای زیربرنامه
  - فعالیتی که توسط زیربرنامه انجام می شود.
- پیاده سازی زیربرنامه شامل:
- پیاده سازی توسط بدنه زیربرنامه تعریف می شود که متشکل از اعلان داده های محلی است
- دستوراتی که عملکرد زیربرنامه را مشخص می کند.

بسته بندی با زیربرنامه ها (ادامه)

تعریف و فراخوانی زیربرنامه

- تعریف زیربرنامه خاصیت ایستای یک برنامه است.
- درحین اجرای برنامه اگر زیربرنامه ای فراخوانی شود سابقه فعالیتی از آن زیربرنامه ایجاد می شود.
- تعریف زیربرنامه قالبی برای ایجاد سابقه فعالیت در حین اجرا است.
- شی داده در حین اجرا برنامه ایجاد می شود:
  - در حین ورود به زیربرنامه
  - توسط عملیاتی مثل malloc

بسته بندی با زیربرنامه ها (ادامه)

تعریف و فراخوانی زیربرنامه(ادامه)

پیاده سازی تعریف و فراخوانی زیربرنامه

- الگو به دو بخش تقسیم می شود:
  - بخش ایستا که سگمنت کد نام دارد و حاوی ثوابت و کد اجرایی است.
  - بخش پویا که رکورد فعالیت نام دارد

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.

تعریف نوع

- 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.

تعریف نوع(ادامه)

هم ارزی نوع

- نوع داده: بتوانیم آن را به طور ایستا تعیین کنیم
- یک موضوع معنایی در تعین مقدار راست شی داده
- تساوی نوع
- 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**

تعریف نوع(ادامه)

هم ارزی نوع(ادامه)

- هم ارزی ساختاری
  - معایب
    - **آیا ترتیب فیلدها باید یکی باشد ...**
    - **دو متغیر ممکن است به طور تصادفی از نظر ساختاری یکسان باشند**
    - **تعیین هم ارزی ساختاری در مورد انواع پیچیده هزینه ترجمه دارد.**
- تساوی اشیای داده
- تساوی پشته
- تساوی مجموعه

تعریف نوع(ادامه)

تعریف انواعی که پارامتردارند

- پیاده سازی: تعریف نوع پارامتردار به عنوان الگویی در زمان ترجمه منظور می شود با این تفاوت که وقتی کامپایلر اعلان یک متغیر را با لیست پارامترهایی که بعد از نام نوع می آید را ترجمه می کند.

فصل هفتم

وراثت

وراثت

- مکانیزمهایی را برای بسته بندی خودکار داده ها توصیف می کنیم
- این مفهوم را طوری بسط می دهیم که عملیات بر روی این اشیا داده از طریق وراثت قابل استفاده باشند.

نگاهی دوباره به انواع داده انتزاعی

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 (ادامه)

- 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 (ادامه)

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

نگاهی دوباره به انواع داده انتزاعی (ادامه)

نمونه سازی تعریف نوع انتزاعی کلی:

- فرایند ایجاد تعریف نوع خاص از تعریف کلی نمونه سازی نام دارد.
- در C++ این مفهوم قالب نام دارد و می تواند برای تولید کلاس کلی به کار رود.
- پیاده سازی: پارامترهای ککیج کلی وقتی که تعریف پکیج نمونه سازی می شود به آن ارسال می گردد.
- خود پکیج به عنوان بخشی از ساختار زمان اجرا وجود ندارد.

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.

وراثت (ادامه)

کلاسهای مشتق

- هر انتزاع شامل توصیفگر داده ها و توابعی است که بر روی اشیایی از آن نوع عمل می کنند(متد)
- تابع همنام کلاس سازنده نام دارد و هنگام ایجاد شی از آن کلاس فراخوانی می شود.
- تابع همنام با کلاس که با ~ شروع می شود مخرب کلاس نام دارد این تابع هنگام از بین رفتن شی از آن کلاس فراخوانی می شود.
- تعریف کلاسی مثل تعریف نوع در C است ولی اعضای تابعی دارد.

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)

کلاسهای مشتق (ادامه)

وراثت چندگانه

- Class A: B,C{…}
- در این اعلان کلاس A از کلاسهای B,C مشتق می شود تا زمانی که مجموعه از اشیای تعریف شده توسط کلاسهای B,C همپوشانی نکنند ادغام آنها برای ایجاد کلاس A مشکلی را به وجود نمی آورد.

وراثت (ادامه)

متدها

- وراثت متدها برای ایجاد اشیای جدید قدرت دیگری اعمال می کند که در بسته بندی موجود نیست.
- برای اشیای کلاس Newstack متد Mytype پیام I am type elemstack را چاپ می کند زیرا تعریف متد ارثی از کلاس elemstack است این مشکل را به دو طریق می توان حل کرد:
- می توانیم متد my type را در تعریف کلاس newstack دو باره تعریف کنیم
- از تابع مجازی استفاده شود.

وراثت (ادامه)

کلاسهای انتزاعی

- گاهی تعریف کلاسها می تواند به صورت یک قابل باشد به طوری که کلاسهای دیگری ازآن ساخته شوند دو روش داریم:
  - ابر کلاسهای انتزاعی
  - وراثت mixin
- 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 هشتم

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:

- ساختارهایی که در عبارات مورد استفاده قرار می گیرند.
- ساختارهایی که بین دستورات یا گروهی از دستورات به کار می روند.
- برنامه نویسی اعلانی
- کنترل ترتیب در برنامه ها

کنترل ترتیب ضمنی و صریح

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.
- مکانیزم کنترل ترتیب در عبارات ترکیب تابعی است یعنی عملیات و عملوندهایش مشخص می شود.
- نمایش درختی ساختارا کنترلی عبارت را نشان می دهد

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

نحو عبارات

- در برنامه ها باید درختها را به صورت خطی مشخص کرد
- نشانه گذاری perfix
- نشانه گذاری Postfix
- نشانه گذاری infix

ترتیب اجرا در عبارات محاسباتی (ادامه)

نمایش درختی عبارات (ادامه)

معنای عبارات

- ارزیابی عبارات perfix
- ارزیابی عبارات Postfix
- ارزیابی عبارات infix
- سلسله مراتب عملگرها (قواعد تقدم عملگرها)
- شرکت پذیری
  - زبان C
  - زبان APL
  - زبان اسمالتاک
  - زبان فورث

ترتیب اجرا در عبارات محاسباتی(ادامه)

نمایش زمان اجرا

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

ترتیب اجرا در عبارات محاسباتی(ادامه)

نمایش زمان اجرا

ارزیابی نمایش درختی عبارت

- مسئله ۱: قواعد ارزیابی یکنواخت
- مسئله ۲: اثرات جانبی
- مسئله ۳: شرایط خطا
- مسئله ۴: عبارات بولین مدار کوتاه

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
  - انتخاب
  - تکرار

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
- امتیاز goto :
  - اگر برچسبها از نظر نحوی ساده باشندمستقیماً توسط سخت افزار پشتیبانی مشود و کارایی آن بالااست
  - 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 دستور ات در متن برنامه لازم نیست با ترتیب اجرا یکی باشد.
  - گروهی از دستورات ممکن است اهداف متعددی داشته باشد.
  - برنامه نویسی ساخت یافته

کنترل ترتیب بین دستورات(ادامه)

کنترل ترتیب ساخت یافته

- دستورات مرکب
  - دستور مرکب
- دستورات شرطی
  - IF
  - ELSE
- دستورات تکرار
  - تکرار ساده
  - تکرار در صورتی که شرط برقرار باشد.
  - تکرار با افزایش یک شمارنده
  - تکرار مبتنی بر دادهها
  - تکرار نامتناهی

کنترل ترتیب بین دستورات(ادامه)

کنترل ترتیب ساخت یافته (ادامه)

- مشکلات کنترل ترتیب ساخت یافته
  - خروج چندگانه از حلقه
  - Do-while-do
  - شرایط استثنایی

کنترل ترتیب بین دستورات(ادامه)

برنامه های بنیادی

- هر فلوچارت حاوی این سه مولفه است:
  - برنامه محض
  - برنامه بنیادی
  - برنامه مرکب
- قضیه ساخت یافته
  - قضیه باهوم و جاکوبینی اثبت کرد که تمام برنامه ها را می توان فقط با ساختارهای کنترلی استاندارد نوشت

ترتیب در عبارات غیر محاسباتی

تطابق الگو

- یک عملیات حیاتی در زبانهایی مثل اسنوبال ، پرولوگ و ام ال تطابق الگو است.
- مجموعه ای از متغیرها به الگوی از پیش تعیین شده انجام می شود.
- بازنویسی ترم
- بازنویسی ترم شکل محدود شده ای از تطابق الگو است که در دامنه زبانهای برنامه سازی کاربردهای فراوانی دارد.

ترتیب در عبارات غیر محاسباتی(ادامه)

اتحاد

- عبارتی حاوی یک یا چند متغیر یک تقاضا نام دارد و رابطه ناشناخته ای را نشان می دهد.
- جانشینی
- اتحاد عمومی
- کاربرد اتحاد در پرولوگ

ترتیب در عبارات غیر محاسباتی(ادامه)

عقبگرد

- اگر به آخرین هدف ممکن برسیم و آن نیز با شکست مواجه شود می گوییم هدف فرعی فعلی شکست خورده است چون مجموعه ای از هدفها در پشته قرار گرفته اند آن را جستجو می کنیم و به هدف فرعی قبلی بر می گردیم که تطبیق صورت گرفته است و تلاش می کنیم هدف دیگری با آن تطابق کند.

ترتیب در عبارات غیر محاسباتی(ادامه)

اصل راه حل

- هدف فضای جستجوی پرولوگ متحد کردن Q1.....Qn است و پرولوگ در انتخاب قاعده ای مثل P از بانک اطلاعاتی آزاد است تا آن را به عنوان فرضیه ای در نظر بگیرد که تفکیک را در آن انجام دهد. اگر با موفقیت انجام شود Б پاسخ به تقاضا را توصیف می کند اگر با شکست مواجه شود نیاز به قاعده P داریم تا جانشین معتبری را بیابد.

فصل نهم

کنترل زیر برنامه

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 .
- 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 این دیدگاه:

- زیربرنامه ها نمی توانند بازگشتی باشند.
- نیاز به دستور فراخوانی صریح است
- زیربرنامه ها در هر فراخوانی باید به طور کامل اجرا شوند
- کنترل به نقطه فراخوانی بر می گردد.
- در هر زمان فقط یک زیربرنامه کنترل را در دست دارد.

کنترل ترتیب زیر برنامه (ادامه)

زیربرنامه های فراخوانی - برگشت

- پیاده سازی: نیاز به چیزهای دیگر:
- بین تعریف زیربرنامه و سابقه فعالیت آن تفاوت وجود دارد.
- سابقه فعالیت دو بخش دارد : سگمنت کد و 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)

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 از P 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.

فصل دهم

مدیریت حافظه

مدیریت حافظه

- با اینکه برنامه نویس با مدیریت حافظه سروکار دارد و باید برنامه هایی بنویسد که از حافظه به خوبی استفاده کند احتمالاً کنترل مستقیم او بر حافظه اندک است.

عناصری که به حافظه نیاز دارند

- سگمنت کد برای برنامه ترجمه شده کاربر
- برنامه های زمان اجرای سیستم
- ثوابت و ساختمان داده های تعریف شده توسط کاربر
- نقاط برگشت زیربرنامه ها
- محیطهای ارجاع
- حافظه های موقت در ارزیابی عبارات
- حافظه های موقت برای انتقال پارامترها
- بافرهای ورودی – خروجی
- داده های خراب سیستم
- عملیات فراخوانی و برگشت از زیربرنامه
- عملیات ایجاد و از بین بردن ساختمان داده ها
- عملیات درج و حذف اجزای ساختمان داده ها

مدیریت حافظه تحت کنترل برنامه نویس و سیستم

مشکل کنترل برنامه نویس بر روی مدیریت حافظه:

- مسئولیت سنگینی را متوجه برنامه نویس می کند و ممکن است با مدیریت حافظه تحت کنترل سیستم تداخل ایجاد کند.
- مراحل مدیریت حافظه
  - تخصیص اولیه
  - بازیابی حافظه
  - فشرده سازی و استفاده مجدد

مدیریت حافظه ایستا

- ساده ترین شکل تخصیص ، تخصیص ایستا است . این تخصیص در زمان ترجمه انجام می شود و در طول اجرا ثابت است.

مدیریت حافظه هرم

- هرم بلوکی از حافظه است که در آن قطعاتی از حافظه به روش غیرساخت یافته تخصیص می یابند و آزاد می شوند.
- تکنیکهای مدیریت حافظه هرم برحسب اینکه اندازه عناصر تخصیص یافته ثابت باشد یا متغیر به دو دسته تقسیم شوند.

مدیریت حافظه هرم (ادامه)

مدیریت حافظه هرم با عناصر طول ثابت

- برای تخصیص یک عنصر اولین عنصر لیست فضای آزاد از لیست حذف می شود و اشاره گری به آن عنصر به عملیات درخواست کننده حافظه برگردانده می شود.
- بازیابی: شمارش ارجاعها و زباله روبی
  - برنامه نویس یا سیستم آنها را بر می گرداند.
  - شمارش ارجاع
  - زباله روبی
    - **نشانه گذاری**
    - **جاروکردن**

مدیریت حافظه هرم (ادامه)

مدیریت حافظه هرم با عناصر طول ثابت (ادامه)

- بخش نشانه گذاری زباله روب کار دشواری است .
- سه فرضیه در مورد این فرآیند نشانه گذاری وجود دارد:
  - هرعنصر فعال باید از طریق زنجیره ای از اشاره گرها با شروع از خارج هرم قابل دستیابی باشد.
  - باید بتوان اشاره گرهای خارج از هرم را که به عنصری در هرم اشاره می کند تعیین کرد
  - باید بتوان در داخل هر عنصر فعال هرم فیلدهایی را تعیین کرد که حاوی اشاره گرهایی به عناصر دیگر هرم اند.

مدیریت حافظه هرم(ادامه)

- مدیریت حافظه هرم با عناصر طول متغیر

تخصیص اولیه و استفاده مجدد

به دلیل متغیربودن طول عناصر دو امکان برای استفاده مجدد وجود دارد:

- استفاده از لیست فضای آزاد برای تخصیص حافظه
- با انتقال تمام عناصر فعال به یک طرف هرم فضای آزاد لیست را فشرده کنید.

مدیریت حافظه هرم(ادامه)

- مدیریت حافظه هرم با عناصر طول متغیر

استفاده مجدد از لیست فضای آزاد

برای مدیریت تخصیص مستقیم از این نوع لیست فضای آزاد چند تکنیک وجود دارد:

- روش اولین جای مناسب
- روش بهترین جای مناسب

مدیریت حافظه هرم(ادامه)

- مدیریت حافظه هرم با عناصر طول متغیر

بازیابی با بلوکهای طول متغیر

- روش برگشت صریح فضای آزاد شده به لیست فضای آزاد ساده ترین تکنیک است اما مسئله های مربوط به زباله ها و ارجاعهای معلق وجود دارد.
- باید تشخیص دهیم که انتهای یک عنصر کجاست وعنصریعدی از کجا شروع می شود.
- ساده ترین راه حل این است که در کنار بیت زباله روبی در اولین کلمه هر بلوک طول آن بلوک نگهداری شود.

مدیریت حافظه هرم(ادامه)

- مدیریت حافظه هرم با عناصر طول متغیر

فشرده سازی و پراکندگی حافظه

- دو روش برای فشرده سازی وجود دارد:
  - فشرده سازی جزئی
  - فشرده سازی کامل