تحلیل و طراحی کمپایلر

حل تمرین سری اول

دانشگاه صنعتی شاهرود

نویسنده : مصطفی فضلی

۱۳۹۸/۱۲/۰۶

استاد دکتر مریم خدابخش

بخش اول مفسر و کامپایلر

مسئله ۱. سوالات نظری

۱. **مزایا و معایب استفاده از مفسر و کامپایلر را نسبت به هم بیان کنید.**

تفاوت بر اساس	کامپایلر	مفسر
مراحل برنامه نویسی	ایجاد برنامه کامپایل تمام بیانیه‌های زبان را برای صحت آن تجزیه و تحلیل می‌کند. اگر نادرست باشد، خطا برمی‌گرداند. اگر خطایی وجود نداشته باشد، کامپایلر سورس کد را به کد ماشین تبدیل خواهد کرد. کامپایلر فایل‌های مختلف کد را به یک برنامه قابل اجرا لینک می‌کند. (مثل exe)	ایجاد برنامه فایل‌ها یا کد ماشین تولید شده را لینک نمی‌کند. بیانیه‌های منبع در زمان اجرا به صورت خط به خط اجرا می‌شوند.
مزیت	کد برنامه از قبل به کد ماشین تبدیل شده است. بنابراین زمان اجرای کد کمتر است.	استفاده از مفسر آسان‌تر است، مخصوصا برای مبتدیان
معایب	شما نمی‌توانید برنامه را بدون بازگشت به کد منبع تغییر دهید.	برنامه‌های تفسیرشده می توانند بر روی کامپیوترهایی که دارای مفسر متناظر هستند اجرا شوند.
کد ماشین زمان اجرا	ذخیره زبان ماشین به عنوان کد ماشین بر روی دیسک کد کامپایل شده سریع‌تر اجرا می‌شود.	کد ماشین را در همه حال ذخیره نمی‌کند. کد تفسیر شده آهسته تر اجرا می‌شود.
مدل	بر اساس مدل بارگیری بر مبنای زبان ترجمه می‌باشد.	بر مبنای روش تفسیر می باشد.
تولید برنامه	برنامه خروجی (در فرمت exe) تولید می‌کند که می‌تواند مستقل ازبرنامه اصلی اجرا شود.	برنامه خروجی تولید نمی‌کند. بنابراین برنامه منبع را هر بار طی اجرا ارزیابی می‌کند.
اجرا	اجرای برنامه از کامپایل جدا است. این کار فقط بعد از اینکه خروجی برنامه کامپایل می‌شود،	اجرای برنامه بخشی از فرآیند تفسیر است. بنابراین خط به خط انجام می‌شود.
حافظه مورد نیاز	برنامه هدف مستقل اجرا می‌شود و نیازی به کامپایلر در حافظه نیست.	مفسر در طی تفسیر در حافظه وجود دارد.
مناسب برای	به ماشین هدف خاصی محدود شده و قابل انتقال نیست. C و C++ محبوب‌ترین زبان‌های برنامه‌نویسی هستند که از مدل کامپایل استفاده می‌کنند.	برای محیط وب، وقتی که زمان بارگیری مهم است. با توجه به تجزیه و تحلیل کاملی که انجام می‌شود، کامپایل‌ها زمان نسبتا بیشتری را برای کامپایل هر کد کوچکی که ممکن است چندین بار اجرا نشوند، می‌گیرند. در چنین مواردی مفسران بهتر می‌باشند.
بهینه‌سازی کد	کامپایلر تمام upfront کد را می‌بیند. از این رو، آن‌ها بهینه‌سازی هایی را انجام می‌دهند که کد را سریع‌تر اجرا می‌کنند.	مفسران خط به خط کد را محاسبه می‌کنند. از این رو، بهینه‌سازی‌ها به اندازه کمپایلر ها قوی نیستند.
تایپ داینامیک کاربرد	پیاده‌سازی برای کامپایلر ها دشوار است چرا که آن‌ها نمیدانند که چه اتفاقی می‌افتد. برای محیط تولید مناسب است.	زبان‌های تفسیری تایپ داینامیک را پشتیبانی می‌کنند. برای برنامه و محیط توسعه مناسب است.
اجرای خطا	کامپایلر تمام خطاها و هشدارها را در زمان کامپایل نشان می‌دهد. بنابراین شما نمی‌توانید برنامه را بدون اصلاح خطاها اجرا کنید.	مفسر تنها بیانیه‌ها را می‌خواند و و خطاها را در صورت وجود نشان می‌دهد. شما باید خطا را برای تفسیر خط بعدی اصلاح کنید.
ورودی	یک برنامه کامل را می‌گیرد.	یک خط از کد را می‌گیرد.
خروجی	کامپایلر کد ماشین واسطه تولید می‌کند.	مفسر هرگز کد ماشین واسطه تولید نمی‌کند.
خطاها	همه خطاها را با هم بعد از کامپایل کردن نشان می‌دهد.	تمام خطاهای هر خط را یک به یک نشان می‌دهد.
زبان‌های برنامه‌نویسی مربوطه	C,C++,C#, Scala, Java همه از کامپایلر استفاده می‌کنند.	PHP, Perl, Ruby از یک مفسر استفاده می‌کنند.

۲. **مزیت استفاده از زبان های میانی چیست؟**

کامپایلر پس از تحلیل معناشناختی یک کد میانجی از کد منبع برای ماشین هدف تولید می‌کند. این کد نمایش‌دهنده برنامه‌ای برای یک ماشین انتزاعی است. این کد میانجی چیزی بین زبان سطح بالا و زبان ماشین است. کد میانجی باید به طرزی ایجاد شده باشد که ترجمه آن به کد ماشین مقصد آسان باشد.

اهداف : بهینه سازی، قابلیت انتقال داده ها و تولید کد انتزاعی مقصد که مستقل از ماشین است.

تمرین سری اول

تحلیل و طراحی سیستم ها

بخش دوم خطاها

مسئله ۲ .خطا پارسر

۱. خطاهاي زير در برنامه هاي يك زبان برنامه سازي متعارف مفروض شده است. كدام دسته از خطاها توسط پارسر قابل كشف است؟ توضيح دهيد.

الف) نابرابري تعداد انديس هاي يك آرايه با تعداد ابعاد تعريف شده آرايه

ب) ناهمخواني نوع متغير A و B در عبارتB+A

ج) نابرابري تعداد پارامترهاي فراخواني يك تابع با تعداد پارامتر هاي تعريف شده براي آن

د) ناهمخواني نوع يك پارامتردر فراخواني يك تابع، با نوع تعريف شده در آن در تعريف تابع

همه گزینه های بیان شده جزو خطاهایی نیستند که پارسر توان کشف آن ها را داشته باشد.

در پارسر درخت توکن ها از روی اطلاعاتی که تحلیلگر لغوی در اختیار تحلیلگر نحوی می گذارد تشکیل می شود، پارسر یا همان تحلیلگر نحوی، وظیفه اعتبارسنجی یک توکن ارائه شده از تحلیل‌گر نحوی را بر عهده دارند.

الف) وظیفه تحلیلگر معنایی است => بررسی تطبیق پذیری سایز آرایه با ابعاد در نظر گرفته شده

ب) وظیفه تحلیلگر معنایی است => بررسی نوع پارامتر ها

ج) وظیفه تحلیلگر معنایی است => بررسی تعداد پارامتر های یک تابع با تعداد تعریف شده

د) وظیفه تحلیلگر معنایی است => بررسی نوع پارامتر های فراخوانی شده

مسئله ۳ .نوع خط

۱. ميدانيم كه هر زيربرنامه ميتواند داراي تعدادي پارامتر باشد و هنگام فراخواني زيربرنامه تعداد آرگومان ها بايد با تعداد پارامترها مطابقت داشته باشد. اگر در برنامه اي اين مطابقت رعايت نشده باشد، خطا مربوطه در كدام يك از مراحل رديابي ميشود؟ توضيح دهيد .

این خطا در مرحله تحلیگر معنایی بررسی می شود. در مرحله بررسی معنایی کامپایلر تعداد و نوع آرگومان های ورودی با تعداد و نوع پارامتر های داده شده قیاس می شوند.

تمرین سری اول

تحلیل و طراحی سیستم ها

بخش سوم کامپایل

مسئله ۴ .تحلیل و ساخت در کامپایل

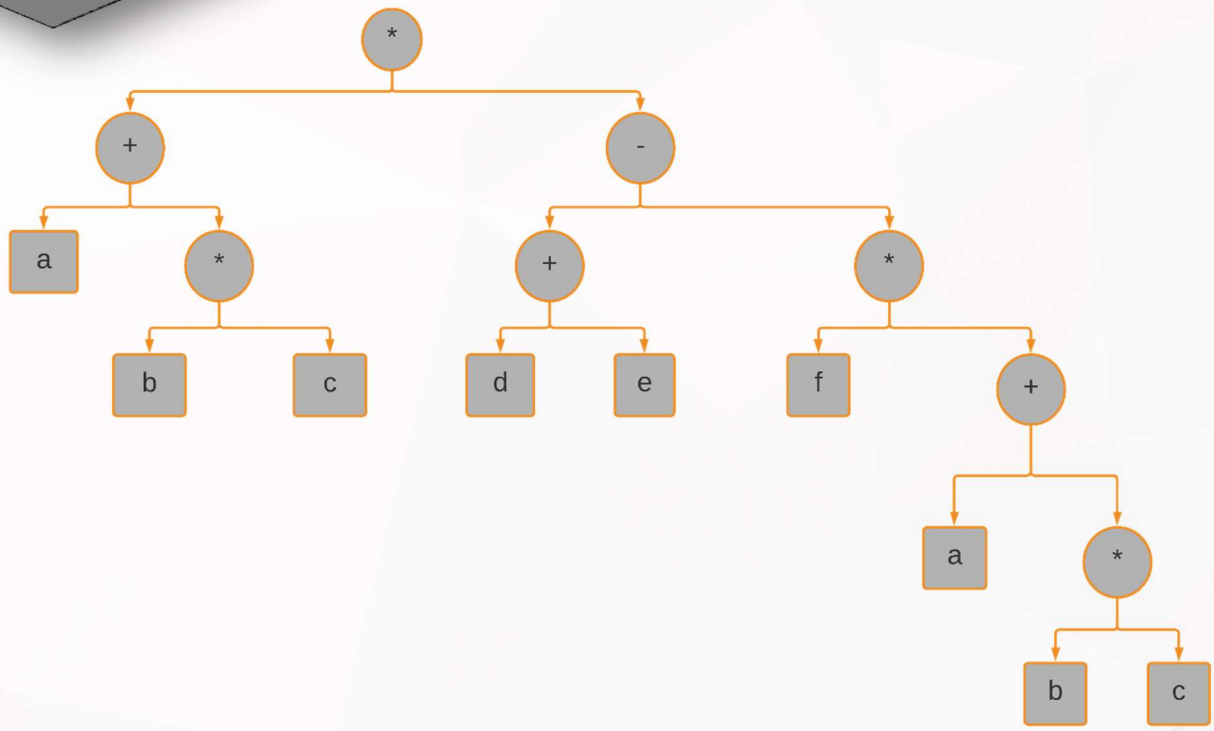
عبارت ((a + b * c) * (d + e – f * (a + b * c) مفروض است .

۱. تمامی نشانه ها و نوع آنها را مشخص کنید.

نوع نشانه	نوع	Variable or function	نام	مشخصه
identifier	Real	variable	a	Id۱
identifier	Real	variable	b	Id۲
identifier	Real	variable	c	Id۳
identifier	Real	variable	d	Id٤
identifier	Real	variable	e	Id٥
identifier	Real	variable	f	Id٦
expression			+	
expression			-	
expression			*	
expression			) و (	

<(> <id1> <+> <id2> <*> <id3> <)> <*> <id4> <+> <id5> <-> <id6> <*> <(> <id1> <+> <id2> <*> <id3> <)> <)>

۲. به کمك پرسش قبلي درخت پارس آن را بدست آورید.



۳. بخش ساخت در فرآیند کامپایل را براي عبارت بالا به صورت کامل تشریح کنید.

در مرحله اول کد میانی با هدف بهینه سازی و قابلیت انتقال کد، به منظور نزدیک تر شدن قالب این کد انتزاعی با زبان ماشین و بهینه سازی راحت تر انجام می شود.

در مرحله دوم بهینه سازی کد و حذف متغیر های میانی غیرضروری به دلیل یافتن راه های بهتر برای اجرای کد و جایگزین کردن کد های تولیدشده با کدهای بهینه صورت می پذیرد.

در مرحله سوم کدی نهایی که کدی قابل جابجایی وابسته به ماشین است تولید می شود.

و در مرحله چهارم و آخه بهینه سازی های جزئی که با دیدی مبتنی بر بستر صورت میگیرد، کدنهایی تولید شده را بهبود می دهد.