پروژه‌ی درس طراحی کامپایلر

زبان Pascal++

دکتر جابری پور

نسخه ۲

۱۱ بهمن ۱۳۹۸

۱ مقدمه

درس طراحی کامپایلرها که یکی از دروس اصلی برنامه درسی رشته‌های علوم کامپیوتر می‌باشد، همواره شامل پروژه‌ای نیز بوده است. عمدتاً دو هدف از طرح پروژه برای این درس وجود دارد؛ آشنایی با طراحی یک زبان برنامه‌سازی و پیاده‌سازی کامپایلر آن و همچنین پیاده‌سازی یک سیستم نرم‌افزاری قابل توجه که توانایی‌های یک مهندس را محک بزنند.

متاسفانه پیاده‌سازی یک زبان برنامه‌سازی واقعی که عموماً مورد استفاده واقع می‌شود بسیار دشوار است. بنابراین لازم است زبانی ساده با قابلیت‌های کافی برا کسب تجارب مورد اشاره طراحی شود.

این پروژه از شما پیاده‌سازی گام به گام یک کامپایلر را می‌خواهد، به این صورت که نیازهای هر بخش را تعریف کرده و از شما می‌خواهد به صورت مجزا آنها را پیاده کنید. شما می‌توانید از زبان‌های C++ ، Java ، Python برای پیاده‌سازی کامپایلر خود بهره ببرید. برای پیاده‌سازی کامپایلر زبان توصیف شده در صورت این پروژه، توصیه می‌شود از ابزاری مانند LLVM استفاده کرد تا روال‌های ساختاریابی و تولید کد ساده‌تر شود. این مورد سر کلاس‌های پروژه بیشتر توضیح داده می‌شود.

۲ ابزارها

برای انجام این پروژه مجاز هستید از هر ابزاری استفاده کنید با این شرط که ورودی برنامه ی شما کدی به زبان Pascal++ می باشد و خروجی آن باید برنامه ی قابل اجرا به فرمت LLVM باشد و قابل اجرا بر روی ماشین مجازی LLVM باشد. البته برای انجام این پروژه ما یکی از دو راهکار زیر را پیشنهاد و آموزش می‌دهیم:

- استفاده از flex/jFlex و یا هر ابزار دیگری برای قسمت Scanner و استفاده از PGen 2.x برای قسمت parser
- استفاده از flex و یا هر ابزار دیگری برای قسمت Scanner و استفاده از Bison برای قسمت parser و استفاده از چارچوب تولید کد LLVM

روش اول دقیقاً مطابق آن چیزی است که در مطالب قبل از میان‌ترم تدریس شد است. نرم‌افزار PGen 2.x به شما کمک می‌کند زبان برنامه‌سازی مورد اشاره را به کمک گراف نحو ساختاریابی کرده و روال‌های مفهومی را پیاده‌سازی کنید. بدیهی است که با انتخاب این روش به سرعت می‌توانید تولید کد پروژه را شروع کنید.

روش دوم بیشتر به بخش دوم درس مبتنی است و نیاز است تا دانش خود را در بخش دوم بالا ببرید. برای استفاده از این روش باید چیزهای بیشتری فرابگیرید اما بعد از یادگیری کار ساده‌تری در پیش دارید اما زمان کمتری نیز دارید.

در این پروژه مجاز نیستید از زبان‌های تابعی علی‌الخصوص Scala و همچنین Parser Combinator استفاده کنید.

۳ بررسی اجمالی

زبان Pascal++ یک زبان Impratrive می باشد (مانند Pascal و یا C++) این زبان شیءگرا نیست. سعی شده این زبان زیاد بزرگ نباشد تا به اتمام رساندن آن مانند یک کابوس نباشد. لازم به ذکر است که

شما می توانید امکاناتی که در زبان تعریف نشده اما زبان های رایج مانند C++ از آن‌ها بهره می‌برند را در این زبان پیاده سازی کنید. این زبان یک تابع main داد که اجرای برنامه از آن شروع می شود. لازم به ذکر است که این زبان Type-Safe می‌باشد.

۴ ویژگی‌های زبان

در این قسمت اجزا و ویژگی‌های زبان را به صورت کلی مرور می‌کنیم. مفاهیم این بخش در بخش ۶ به صورت دقیق تعریف می‌شوند.

همانطور که در بخش قبل اشاره شد، این زبان شامل تعدادی تابع و ساختار است. در اینجا این موارد را به صورت غیر رسمی توضیح می‌دهیم.

۱.۴ تابع

توابع این زبان متشکل از تعدادی ورودی و تعدادی خروجی می باشند. تعریف تابع به زبان Pascal مشاهبت دارد.

```
function <function_name>( arg1: type1; arg2: type2, ...) : return_type
begin
   // function body
   return 1 ;
end

procedure <procedure_name>( arg1: type1; arg2: type2, ...)
begin
    // procedure body
end
```

لازم به ذکر است که تابعی که خروجی نداشته باشد `procedure` می باشد ارسال تمامی آرگومان‌ها به هر تابعی، به طور پیش‌فرض به شکل Call By Value [۱] است.

۲.۴ انواع

انواع داده های این زبان به شرح زیر می باشد.

۱. boolean مقادیر بولی، ۱بایت، شامل مقادیر ۰ و ۱.

۲. integer اعداد صحیح، ۴بایت، ذخیره‌سازی به شیوه مکمل ۲.

۳. character کاراکتر، ۱بایت، ذخیره‌سازی به شیوه اعداد صحیح.

۴. real اعداد حقیقی، ۴ بایت، ذخیره‌سازی به شیوه IEEE754 با دقت معمولی.

۵. string رشته، حاوی تعدادی کاراکتر، شیوه ذخیره‌سازی: به تشخیص کامپایلر نویس!

[۱] یعنی تغییرات در بدنه تابع روی متغیرها اثری ندارد.

کلیه انواع ابتدایی (بجز رشته) به یکدیگر قابل تبدل می‌باشند. در تبدیل انواع صحیح ،کاراکتر و اعشاری به بولی و بالعکس، عدد ذخیره شده به شکل ۰ یا ۱ خواهد بود. در تبدیل اعداد اعشاری به انواع صحیح و کارکتر نیز، بخش اعشاری عدد حذف شده و بخش کم ارزش قسمت صحیح عدد اعشاری، براساس میزان فضای نوع مقصد، در آن ذخیره می‌شود. در تبدیل عکس نیز عدد باید به نزدیکترین عدد اعشاری ممکن تبدیل شود. تبدیل انواع به یکدیگر به صورت اتوماتیک و توسط کامپایلر انجام می شود.

۳.۴ توابع تعبیه شده

مانند بسیاری از زبان‌های برنامه‌سازی، این زبان شامل تعدادی تابع است. این توابع تعریف نمی‌شوند، بلکه از پیش در زبان موجود هستند.

- main شروع برنامه از این تابع است. این تابع با سایر توابع دیگری که در این لیست می‌آید متفاوت است. زیرا برنامه نویس خودش این تابع را تعریف می‌کند:

```
function main() : integer
begin
  // function body
end
```

- IO توابع ورودی‌ـخروجی زبان. دو تابع از این نوع وجود دارد:
  - `read(id)`
    این تابع، بر اساس نوع ورودی، آن نوع را از ورودی خوانده و داخل حافظه ارسال شده قرار می‌دهد. نوع ورودی جزء انواع پایه‌ای است. (به بخش ۲.۴ مراجعه کنید)
  - `write(expr)`
    یک عبارت ارزیابی شده و مقدار آن در خروجی استاندارد نوشته می‌شود.
- توابع رشته
  - `strlen(string): integer`
    این تابع طول یک رشته را در قالب یک عدد صحیح باز می‌گرداند.

۴.۴ دستورالعمل‌ها و عبارات

در این بخش، قسمت اصلی زبان یعنی عبارات آن را معرفی می‌کنیم.

۱.۴.۴ ثوابت

ساده‌ترین نوع عبارات هستند:

- ثوابت بولی مثل true و false
- ثوابت عددی، از جمله ثوابت صحیح و ثوابت حقیقی
- ثوابت کارارکتری مثل `'c'`
- ثوابت رشته‌ای مانند `"this is a string"`

۲.۴.۴ شناسه‌ها

نام متغیرهای محلی، نام پارامترهای فرمال تابع و نام عناصر ساختار جزء شناسه‌ها هستند. همچنین برای معرفی نام توابع و ساختارها هم از شناسه استفاده می‌شود، ولی توابع و ساختارها از این جهت جزء عبارات نیستند. این شناسه‌ها نباید با کلمات کلیدی زبان تداخل داشته باشند.

۳.۴.۴ انتساب

انتساب به فرم `id := expr ;` می‌باشد. در انتساب در صورت لزوم عمل تبدیل نوع برای انواع پایه، صورت می‌پذیرد. مقدار این عبارت در صورتی تبدیل نوع رخ ندهد ۱ بوده و درغیر اینصورت ۰ خواهد بود. همچنین همانطور که در بخش ۵.۴ آمده به کمک کلید واژه‌ای خاص می‌توان به ابعاد آرایه مقدار دهی کرد.

۴.۴.۴ فراخوانی تابع

در این زبان توابع به شکل `id(expr, expr, ....)` فراخوانی می‌شوند. در این عبارت، آرگومان‌ها از چپ به راست مورد ارزیابی قرار می‌گیرند. همچنین مقدار این عبارت برابر خروجی تابع خواهد بود.

۵.۴.۴ عبارات شرطی

عبارات شرطی در زبان به صورت

```
if expr then
begin
   // code
end
else
begin
   // code
end
```

است. معنا و تفسیر این عبارت، به شکل استاندارد است. یادآوری می‌شود که بخش `else` تنها در صورتی که مقدار عبارت برابر ۰ یا `false` باشد اجرا می‌شود. این عبارت مقدار بازگشتی ندارد.

۶.۴.۴ حلقه‌ها

در این زبان یک نوع حلقه وجود دارد:

`while` این حلقه به شکل زیر نوشته می‌شود:

```
while (expr) do
begin
  //code
end
```

۷.۴.۴ قطعه

یک قطعه فرم زیر را دارد:

```
begin
 stmt ; stmt ; ..
end
```

یعنی یک قطعه حاوی تعدادی عبارت است، این عبارات به ترتیب مورد ارزیابی قرار می‌گیرند.

۸.۴.۴ تعریف متغیر

تعریف متغیر به شکل زیر است:

```
a : type := expr ;
```

در این ساختار استفاده از یک عبارت جهت انتساب اختیاری است. در اینجا باید کنترل شود که مقدار عبارت قابل انتساب به نوع مربوطه باشد.

۹.۴.۴ عملگر حسابی و مقایسه‌ای

این زبان شامل تعدادی عملگر دوگانی به شرح زیر است:

- Add: $+$
- Subtract: $-$
- Multiply: $*$
- Divide: /
- Bitwise AND: &
- Exclusive Add: ^
- Bitwise OR: |
- Logical AND: and
- Logical OR: or
- Mod: %

لازم به ذکر است که عملگرهای بیتی فقط بر روی انواع صحیح قابل تعریف‌اند.
همچنین عملگرهای یگانی این زبان شامل:

- Minus: $-$
- Logical Not: ~

است.

همه عملگرهای غیر منطقی بالا (که مقدار بولین باز می‌گردانند) بسته به نوع عملوندشان (صحیح یا حقیقی) نوع بازگشتی‌شان تعیین می‌شود.

این زبان عملگرهای مقایسه‌ای زیر را دارد:

- Less than: $<$
- Less or Equal: $>=$
- Equal: $=$
- Not Equal: $<>$
- Greater or Equal: $<=$
- Greater than: $>$

این عملگرها مقدار بولین باز می‌گردانند. مابقی عملگرها بسته به نوع مقدار آن نوع را باز می‌گردانند.

۵.۴ آرایه‌ها

در این زبان می‌توان آرایه‌های چند بعدی را تعریف کرد. این تعریف، با کمی تغییر بسیار شبیه به زبان Pascal است.

```
arrID : array [expr] of type
```

۵ ساختار واژگانی

واحدهای واژگانی این زبان متشکل از اعداد، صحیح و حقیقی، شناسه‌ها، واژه‌های کلیدی و رشته‌ها می‌شود. همچنین این زبان حساس به کوچک و بزرگ بودن حروف است.

۱.۵ ثوابت و شناسه‌ها

۱.۱.۵ اعداد صحیح

هر رشته کاراکتری از ارقام (۰ تا ۹) که ناتهی باشد را یک عدد صحیح می‌نامیم. این اعداد به صورت پیش‌فرض عدد صحیح ۴بایتی در نظر گرفته می‌شوند. در غیر این صورت، تعریف چنین اعدادی خطا محسوب می شود. همچنین اعداد، در مبنای ۱۰ در نظر گرفته می‌شوند، مگر آنکه قبل از رشته اعداد صحیح 0x بیاید، که به معنای آن است که عدد در مبنای ۱۶ نوشته شده.

۲.۱.۵ اعداد حقیقی

هر رشته ناتهی از ارقام که شامل کاراکتر نقطه(.) باشد را یک عدد حقیقی در نظر می‌گیریم.

۳.۱.۵ کاراکاتر

یک کاراکتر! جهت استفاده به صورت ثابت، داخل Single Quotation قرار می‌گیرد.

۴.۱.۵ رشته

رشته‌ها داخل Double Quotation محصور می‌شوند. نحوه نوشتن کاراکترهای خاص ثوابت رشته‌ای مانند زبان Pascal است.

۵.۱.۵ شناسه

شناسه‌ها با حروف الفبای انگلسی شروع می‌شوند و ترکیبی از عدد، شناسه و زیرخطUnderscore می‌باشند.

۲.۵ Comment

۱.۲.۵ تک خطی

هرگاه دو علامت خط تیره ((Dash(-) به شکل متوالی بیایند، خط جاری، از نقطه وقوع تا انتها Comment حساب می‌شود.

۲.۲.۵ چند خطی

هر آنچه بین --> و <-- قرار گیرد (اعم از خط و کاراکتر) به عنوان Comment محسوب می‌شود.

لطفاً به این مثال توجه کنید:

```
--this is single line comment
// this line is also comment
<-- a multiple
  line comment -->
```

۳.۵ White Space

کاراکترهای زیر، فاصله سفید محسوب می‌شوند:

- Blank, ASCII 32
- \n, ASCII 10
- \f, ASCII 12
- \r, ASCII 13
- \t, ASCII 9
- \v, ASCII 11

۴.۵ کلمات کلیدی

کلمات کلیدی این زبان به شرح زیر است:

array, assign, boolean, break, begin, char, continue, do, else, end, false, function, procedure, if, integer, of, real, return, string, true, while, var

۶ نحو زبان Pascal++

در این بخش در حد مناسبی آنچه تا کنون درباره زبان گفتیم را به شکل رسمی عنوان می‌کنیم.

۱.۶ گرامر زبان

شاید این قسمت مهم‌ترین بخش این مستند باشد؛ قواعد تولید زبان به شکل BNF آورده شده‌اند. البته به تمام قواعد BNF پایبند نبودیم. برای سادگی درک گرامر، به نوعی از عبارات منظم هم استفاده شده است. به طور مشخص A^* یعنی صفر یا بیشتر تکرار متوالی A، A^+ یعنی یک یا بیشتر تکرار متوالی A. همه علائمی که به داخل [...] قرار داشته باشند، اختیاری هستند. جهت گروه‌بندی و خوانایی بیشتر در گرامر از علامت ⟦...⟧ استفاده شده که برای مشخص نمودن نمادهای مرتبط است. همچنین علامت + به معنای جمع در عبارات منظم است. هرجا منظور جمع بوده به شکل '+' آمده است. در مورد bracket هم این موضوع صادق است. در این گرامر number به منظور عدد صحیح ثابت است

⟨ PROGRAM ⟩	::=	⟦ ⟨ FUNC_DEF ⟩ + ⟨ PORC_DEF ⟩ + ⟨ VAR_DCL ⟩ ⟧*
⟨ VAR_DCL ⟩	::=	⟨ SIMPLE_VAR ⟩
	|	⟨ ARRAY_VAR ⟩
⟨ SIMPLE_VAR ⟩	::=	*id* : ⟨ TYPE ⟩ [⟨ ASSIGNMENT ⟩]
⟨ ARRAY_VAR ⟩	::=	*id* : *array* ⟨ TYPE ⟩ *of* ⟨ ARRAY_DIMENSION ⟩
⟨ ARRAY_DIMENSION ⟩	::=	[*number* ⟦ ,number ⟧*]
⟨ ASSIGNMENT ⟩	::=	:= ⟨ EXPR ⟩
⟨ BULK_ASSIGNMENT ⟩	::=	(⟨ EXPR ⟩ ⟦ , ⟨ EXPR ⟩ ⟧*)
⟨ FUNC_DEF ⟩	::=	*function id* () : ⟨ TYPE ⟩ ⟨ BLOCK ⟩
	|	*function id* (⟨ ARGUMENT ⟩) : ⟨ TYPE ⟩ ⟨ BLOCK ⟩
⟨ PROC_DEF ⟩	::=	*procedure id* () ⟨ BLOCK ⟩
	|	*procedure id* (⟨ ARGUMENT ⟩) ⟨ BLOCK ⟩
⟨ ARGUMENT ⟩	::=	⟨ VAR_DCL ⟩ ⟦ ; ⟨ VAR_DCL ⟩ ⟧*
⟨ BLOCK ⟩	::=	*begin* ⟦ ⟨ STMT ⟩; ⟧* *end*
⟨ STMT ⟩	::=	⟨ FUNCTION_CALL ⟩
	|	(⟨ ID ⟩ ⟦ , ⟨ ID ⟩ ⟧+) ⟨ BULK_ASSIGNMENT ⟩
	|	⟨ ID ⟩ ⟨ ASSIGNMENT ⟩

	|	⟨ VAR_DCL ⟩
	|	⟨ LOOP ⟩
	|	*return id*
	|	⟨ CONDTIONAL ⟩
⟨ EXPR ⟩	::=	(⟨ EXPR ⟩)
	|	⟨ EXPR ⟩ ⟨ BINARY_OP ⟩ ⟨ EXPR ⟩
	|	⟨ UNARY_OP ⟩ ⟨ EXPR ⟩
	|	⟨ ID ⟩
	|	⟨ CONSTANT ⟩
	|	⟨ FUNCTION_CALL ⟩
⟨ FUNCTION_CALL ⟩	::=	*id*([⟨ EXPR ⟩ ⟦ , ⟨ EXPR ⟩ ⟧*])
⟨ ID ⟩	::=	*id*
	|	*id* [⟨ EXPR ⟩ ⟦ ,⟨ EXPR ⟩ ⟧*]
⟨ UNARY_OP ⟩	::=	See Section ۹.۴.۴
⟨ BINARY_OP ⟩	::=	See Section ۹.۴.۴
⟨ CONSTANT ⟩	::=	See Section ۱.۵
⟨ LOOP ⟩	::=	*while* (⟨ EXPR ⟩) *do* ⟨ BLOCK ⟩
⟨ CONDITIONAL ⟩	::=	*if* (⟨ EXPR ⟩) *then* ⟨ BLOCK ⟩ [*else* ⟨ BLOCK ⟩]

۲.۶ تقدم عملگرها

تقدم علمگرهای بخش ۹.۴.۴ مشابه زبان C می‌باشد. برای اطلاعات بیشتر می‌توانید به اینجا مراجعه کنید.

۷ محیط زمان اجرا

برنامه باید تحت ماشین مجازی LLVM اجرا شود. این ماشین مجازی در اختیار شما قرار خواهد گرفت. لازم به ذکر است که دستورات این ماشین در کلاس پروژه برای شما توضیح داده خواهد شد.

۸ انجام و تحویل پروژه

پروژه باید در قالب گروه‌های یک یا دو نفره انجام شود. توجه کنید که گروه تک نفره هیچ مزیتی بر گروه دو نفره از نظر نمره ندارد بنابراین عدم انجام کار گروهی جهت دریافت نمره بیشتر نتیجه بخش نخواهد بود. همچنین گروه‌ها باید از طریق این فرم تا تاریخ ۲۰ آبان ماه اعلام گردند. عدم اعلام گروه تا تاریخ

فوق به منزله انصراف از انجام پروژه می‌باشد. کلیه پروژه‌ها باید تا تاریخ ۱۱ بهمن ۹۸ تکمیل شده و ارسال گردند (از طریق فرم تحویل).

پس از تحویل پروژه ها به هر گروه زمانی جهت تحویل حضوری تخصیص می‌یابد که طی این زمان باید کد ارسال شده را کامپایل کرده و تست‌ها را نیز اجرا کنند. تغییر کد ارسال شده در زمان تحویل مجاز است اما زمان تحویل هر گروه محدود به ۴۵ دقیقه برای اجرای همه تست‌هاست و پس از ۴۵ دقیقه تعدادی سوال جهت ارزیابی میزان تسلط اعضای گروه پرسیده می‌شود. لازم به ذکر است که همه اعضای گروه باید از چگونگی پیاده‌سازی همه بخش‌های پروژه مطلع باشند به گونه‌ای که گویی خود آن بخش را پیاده‌سازی کرده‌اند. همچنین لازم به ذکر است که در صورتی که یک تست را به درستی پاسخ دهید، ۱۰ درصد از نمره پروژه را دریافت خواهید کرد. تست‌ها حاوی یک برنامه به زبانی که در این سند توصیف شده می‌باشند که باید آن را به کد llvm تبدیل کنید. سپس به فایل خروجی شما تعداد ورودی و خروجی داده می‌شود و صحت عملکرد کامپایلر شما مبتنی بر صحت برنامه تبدیلی شما سنجیده خواهد شد. بدهی است که برخی از بخش‌های پروژه پیش‌نیاز سایر بخش‌ها هستند و به همین جهت با تکمیل اجرای یک تست بخشی از نمره پروژه را دریافت خواهید کرد. با توجه به اینکه زمان تحویل پروژه شما محدود است از ابتدا مواردی را روی پروژه‌تان بیازمایید که آمادگی آن را دارید و زمان تغییر را برای باگ‌های احتمالی ذخیره کنید.

همچنین توجه کنید که موارد این صورت پروژه ممکن است حاوی اشکالاتی باشند. جهت اطمینان، هرگونه موردی را با گروه حل‌تمرین در میان گذاشته تا اشکال بلافاصله برطرف شود. همچنین پیگیر تغییرات و نسخه‌های آتی صورت پروژه از طریق پیازا باشید.

۹ نمرات قسمت‌های مختلف

در جدول زیر لیست قسمت‌های اجباری و امتیازی آمده است. لازم به ذکر است که برخی از قسمت‌ها با وجود امتیاز پایین، در بقیه تاثیر دارند. برای مثال در صورت پیاده‌سازی نکردن قسمت انواع، ممکن نیست که نمره‌ی بخش محاسبات به شما تعلق گیرد. در جدول زیر موارد امتیازی با ستاره مشخص شده‌اند. همچنین مواردی که برای دریافت نمره باید پیاده‌سازی شوند با علامت تعجب! مشخص شده‌اند. بدون پیاده‌سازی این موارد نمره‌ای دریافت نخواهید کرد. بعلاوه مجدداً توجه کنید که موارد زیر ۹۰ درصد نمره شما را تشکیل می‌دهند و ۱۰ درصد دیگر را با اجرای تنها یک تست خواهید گرفت. همچنین این جدول ممکن است پیش از تحویل نهایی کمی دچار تغییر شود.

مورد تولید کد	ضریب نسبی	امتیازی/اجباری
خواندن از ورودی	۲	!
نوشتن در خروجی	۲	!
پیاده سازی انواع کاراکتری و بولی	۱	
پیاده سازی نوع اعداد حقیقی	۱	
پیاده سازی نوع رشته	۳	
پیاده سازی نوع long	۱	
انتساب صحیح داده های هم نوع (به جز رشته)	۲	
انتساب صحیح رشته	۲	
انتساب صحیح داده های غیر هم نوع	۳	
محاسبات جمع و تفریق اعداد صحیح	۲	
محاسبات ضرب و تقسیم اعداد صحیح	۲	
رعایت تقدم عملگرهای محاسباتی	۲	
محاسبات بیتی	۲	
پیاده سازی انواع آرایه	۵	
پیاده سازی انتساب رشته به آرایه کارکتری	۳	
پیاده سازی توابع رشته‌ای تعبیه شده strlen	۲	
محاسبات ممیز شناور	۱	
محاسبات ترکیبی انواع صحیح کاراکتری و حقیقی	۴	
پیاده سازی ساختار شرطی	۵	
پیاده سازی قطعه و تعیین درست محدوده عمر متغیرهای داخل آن	۳	
پیاده سازی متغیر های سراسری	۱	*
پیاده سازی حلقه	۴	
پیاده سازی تابع main	۱	!
پیاده‌سازی توابع (فراخوانی، اجرا (برای هر دو نوع procedure و function))	۵	*
پیاده سازی تابع بازگشتی	۲	*
پیاده سازی مقدار بازگشتی توابع	۱	*
عملکرد صحیح جدول نماد ها	۳	
Bulk Assignment	۲	*

۱۰ تغییرات

تغییرات زیر در پروژه اعمال شده است:

- برخی اشکالات تایپی در بخش‌های ۷.۴.۴ و ۸.۴.۴.

- در قسمت ۲.۵ مثال مدل دیگر کامنت هم افزوده شد.
- در بخش گرامرها با توجه به مشکلاتی که وجود داشت اصلاحات زیر انجام شد:
  - بخش‌های $Simple\_var$ و $Array\_var$ اصلاح شد.
  - $array\_dimention$ جهت برطرف کردن نیاز به استفاده از تخصیص حافظه پویا افزوده شد.
  - انتساب تجمیعی و عادی تفکیک شده و انتساب تجمیعی ببخش‌های امتیازی افزوده شد و انتساب تو در تو حذف شد.
  - بخش‌های $func\_def$ و $proc\_def$ به منظور اضافه شدن جداکننده میان آرگومان‌ها تغییر کرد.
  - تعریف id اصلاح شد. (تعریف قبلی بی معنا بود)
  - بخش آخر $expr$ به انتساب تجمیعی منتقل شد.
- کژتابی نگارشی بخش ۲.۴ برطرف شد.
- اصلاح جدول امتیازات و افزودن انتساب تجمیعی