بسمه تعالی

پروژه پایانی

فاز سوم ـ پیاده‌سازی تحلیل‌گر معنایی و تولیدکننده کد میانی

(مهلت ارسال، ۹ تیر ساعت ۸:۰۰)

شما در فازهای قبلی، یک اسکنر و پارسر برای بخشی از زبان C پیاده‌سازی کردید. در این فاز، باید یک تحلیل‌گر معنایی و یک تولیدکننده کد میانی به اسکنر و پارسری که پیاده‌سازی کردید، اضافه کنید. توجه کنید که استفاده از کدهای موجود در مرجع درس یا سایر کتب کامپایلر، در صورت تسلط بر آن کد و اعلام مأخذ در مستندات همراه پروژه اشکالی ندارد ولی استفاده از کدها و برنامه‌های موجود در سایت‌ها و کدهای سایر گروه‌ها (در همین نیم‌سال یا سال‌های گذشته) اکیدا ممنوع است و در اکثر موارد سبب مردودی در درس خواهد شد. در این مورد تفاوتی میان گروه دهنده یا گیرنده کد وجود ندارد. دقت کنید این گزاره حتی در صورتی که فازهای قبل را پیاده‌سازی نکرده‌اید نیز، درست است و شما نمی‌توانید از اسکنر یا پارسر سایر گروه‌ها استفاده کنید و باید اسکنر و پارسر مورد نیازتان را خودتان، و طبق توضیحاتی که در سندهای فازهای قبلی آمده، پیاده‌سازی کنید.

گرامر NC-Minus

در صورتی که در فاز دوم، پارسر را پیاده‌سازی نکرده‌اید، در نظر داشته باشید که عملیات پارس باید طبق این گرامر صورت بپذیرد. در این گرامر، پایانه‌ها پررنگ‌تر از غیر پایانه‌ها نمایش داده شده‌اند. دقت کنید شما نباید تغییراتی در گرامر اعمال کنید که باعث تغییر زبان شود. بنابراین هر گونه تغییری که در گرامر برای تبدیل آن به فرم LL(1) می‌دهید، نباید زبان آن را تغییر دهد.
چه در حالتی که گرامر را در فاز دوم تغییر داده‌اید و یا در این فاز تغییر داده‌اید، دقت کنید که پیاده‌سازی کامپایلر باید با استفاده از گرامر تغییر داده‌ی تیم خودتان صورت بگیرد و در فایل‌های نهایی تحویلی پروژه، یک فایل شامل گرامر تغییر داده شده نیز قرار دهید.

1. program → declaration-list **EOF**
2. declaration-list → declaration-list declaration | ϵ
3. declaration → var-declaration | fun-declaration
4. var-declaration → type-specifier **ID ;** | type-specifier **ID** [**NUM**] **;**
5. type-specifier → **int** | **void**
6. fun-declaration → type-specifier **ID** (params) compound-stmt
7. params → param-list | **void**
8. param-list → param-list **,** param | param
9. param → type-specifier **ID** | type-specifier **ID** [ ]
10. compound-stmt → **{** declaration-list statement-list **}**
11. statement-list → statement-list statement | ϵ
12. statement → expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt | switch-stmt
13. expression-stmt → expression **;** | **continue ;** | **break ;** | **;**
14. selection-stmt → **if** (expression) statement **else** statement
15. iteration-stmt → **while** (expression) statement
16. return-stmt → **return ;** | **return** expression **;**
17. switch-stmt → **switch** (expression) **{** case-stmts default-stmt **}**

18. case-stmts → case-stmts case-stmt | ϵ

19. case-stmt → **case NUM :** statement-list

20. default-stmt → **default :** statement-list | ϵ

21. expression → var **=** expression | simple-expression

22. var → **ID** | **ID** [expression]

23. simple-expression → additive-expression relop additive-expression | additive-expression

24. relop → **<** | **==**

25. additive-expression → additive-expression addop term | term

26. addop → **+** | **-**

27. term → term ***** signed-factor | signed-factor

28. signed-factor → factor | **+** factor | **-** factor

29. factor → **(** expression **)** | var | call | **NUM**

30. call → **ID** **(** args **)**

31. args → arg-list | ϵ

32. arg-list → arg-list **,** expression | expression

تحلیل‌گر معنایی و تولیدکننده کد میانی

شما در این فاز باید تحلیل‌گر معنایی و تولیدکننده کد میانی را پیاده‌سازی کنید. به این منظور در زمان پارس کردن و حرکت روی دیاگرام پارس، هنگام پیمایش برخی از یال‌ها، بایستی تحلیل‌گر معنایی و یا تولیدکننده کد میانی توسط پارسر فراخوانی شده تا روتین معنایی لازم (روتین تحلیل معنایی یا تولید کد میانی) اجرا شود. به عبارت دیگر، این فعالیت‌ها به صورت همزمان (Pipeline) با فعالیت‌های پارسر و اسکنر انجام می‌شوند و کامپایلر تنها با یک گذر (Pass) همه وظایف مربوط به تحلیل لغوی، نحوی، معنایی و تولید کد میانی را انجام می‌دهد. دقت کنید بخشی از نمره‌ی این فاز، مربوط به درست کار کردن بخش‌های قبلی و درست روی هم سوار شدن اجزای کامپایلر می‌باشد، بنابراین از درستی کارکرد اسکنر و پارسر خود، اطمینان حاصل کنید.

نکات مربوط به تحلیل‌گر معنایی و مولد کد میانی

- همانطور که از گرامر زبان مشخص است، برنامه شامل تعدادی تعریف متغیر و تابع می‌باشد. این تعریف‌ها با هر ترتیبی می‌توانند ظاهر شوند. دقت کنید که با توجه به تک گذره بودن کامپایلر، در هر بخش از کد، تنها می‌توان از توابعی که در scope فعلی یا بالاتر و در خط‌های قبل تر برنامه تعریف شده‌اند، استفاده کرد.

- هر برنامه در این زبان حتما باید شامل تعریفی برای تابع main به صورت global و با signature زیر باشد:

```
void main(void);
```

در صورتی که فایل ورودی فاقد تعریفی برای main باشد و یا signature آن، با چیزی که بالا گفته شده همخوانی نداشته باشد، باید پیغامی به صورت زیر بدهید:

```
main function not found!
```

در تست های بدون خطا، همواره تابع main ، آخرین تعریف global در برنامه خواهد بود.

- برنامه‌ی شما باید به صورت ضمنی، شامل تعریفی برای تابعی به نام output با signature زیر باشد:

```
void output(int a);
```

با فراخوانی این تابع، باید مقداری که به عنوان ورودی تابع داده می‌شود، در خروجی برنامه چاپ شود.

- همانطور که در این سند و سند قبلی گفته شده است، شما حق اعمال تغییراتی در گرامر زبان را که باعث تغییر زبان خروجی آن بشود، ندارید. بنابراین برای پیاده‌سازی موارد بالا، نمی‌توانید از راه‌حل اعمال تغییرات در گرامر زبان، استفاده کنید.

- در ساختارهای شرطی مانند if و while در صورتی که حاصل عبارت expression صفر باشد، عبارت false در نظر گرفته می‌شود و در غیر این صورت true در نظر گرفته می‌شود. همچنین حاصل اعمال relop در یک عبارت، در صورتی که نتیجه true باشد، عدد یک خواهد بود و در صورتی که نتیجه false باشد، عدد صفر خواهد بود.

- در ساختار switch در صورت نبودن دستور break ، دستورات case های بعدی نیز اجرا می‌شوند.

- در فراخوانی توابع، مقادیر int باید به صورت مقدار [1] و مقادیر آرایه باید به صورت مرجع [2] به تابع داده شوند.

---

[1] Pass by value
[2] Pass by reference

خطاهای معنایی

در ادامه خطاهای معنایی آورده شده است که پیاده‌سازی آنها اجباری است. پیاده‌سازی سایر خطاهای معنایی احتمالی که در این لیست قرار ندارند، کاملا اختیاری است و تست‌های داده شده برای ارزیابی کد نهایی پروژه فقط شامل خطاهای معنایی زیر خواهند بود:

۱. Scoping:

هر متغیر یا تابع، قبل از دسترسی باید در scope جاری یا در یکی از scope هایی که scope جاری را در بر دارد، تعریف شده باشد. چنانچه برای یک متغیر یا تابع با نام ID این موضوع صادق نبود، پیغامی به صورت زیر بدهید:

'ID' is not defined.

۲. void بودن نوع متغیر:

در هنگام تعریف یک متغیر تکی یا یک آرایه نمی‌توان نوع آن را void قرار داد. در صورت بروز این اتفاق پیغامی به صورت زیر بدهید:

Illegal type of void.

۳. تطابق تعداد ورودی توابع:

تعداد متغیرهایی که هنگام فراخوانی به یک تابع می‌دهیم، باید با تعداد آرگومان‌های ورودی بر اساس تعریف آن تابع مطابق باشد. در صورت مغایرت اگر نام تابع ID بود باید پیغامی به صورت زیر بدهید:

Mismatch in numbers of arguments of 'ID'.

۴. بررسی محل قرارگیری continue:

چنانچه بیرون از حلقه while عبارت continue آمده باشد، باید پیغامی به صورت زیر بدهید:

No 'while' found for 'continue'.

۵. بررسی محل قرارگیری break:

چنانچه عبارت break بیرون از حلقه while یا بیرون از ساختار switch آمده باشد، باید پیغامی به صورت زیر بدهید:

No 'while' or 'switch' found for 'break'.

۶. هم‌نوع بودن عملوندها:

در هنگام اعمال هر یک از عملگرهای جمع و ضرب و مقایسه می‌بایست بررسی کرد که آیا عملوندهای دو طرف از نوع عددی هستند یا نه. در صورتی که از این نبودند باید پیغامی به صورت زیر بدهید:

Type mismatch in operands.

فهرست دستورالعمل‌های سه آدرسه قابل استفاده برای تولید کد میانی

توجه داشته باشید که قالب کد میانی که تولید می‌کنید باید مطابق با جدول زیر باشد:

قالب کد سه‌آدرسه	توضیح
(ADD, S1, S2, D)	عملوندهای اول و دوم جمع می‌شوند و حاصل در D قرار می‌گیرد.
(SUB, S1, S2, D)	عملوند دوم از عملوند اول کم می‌شود و حاصل در D قرار می‌گیرد.
(AND, S1, S2, D)	عملوندهای اول و دوم AND می‌شوند و حاصل در D قرار می‌گیرد.
(ASSIGN, S, D,)	محتوای S در D قرار می‌گیرد.
(EQ, S1, S2, D)	اگر S1 و S2 مساوی باشند، در D مقدار true و در غیر این صورت، مقدار false ذخیره می‌شود.
(JPF, S, L,)	محتوای S بررسی می‌شود و در صورتی که false باشد، کنترل به L منتقل می‌شود.
(JP, L, ,)	کنترل به L منتقل می‌شود.
(LT, S1, S2, D)	اگر S1 کوچکتر از S2 باشد، در D مقدار true و در غیر این صورت، مقدار false ذخیره می‌شود.
(MULT, S1, S2, D)	عملوند اول در عملوند دوم ضرب می‌شود و حاصل در D قرار می‌گیرد.
(NOT, S, D,)	نقیض محتوای عملوند S در D قرار می‌گیرد.
(PRINT, S, ,)	محتوای S بر روی صفحه چاپ می‌شود.

- از روش‌های نشانی‌دهی [۳] مستقیم (مانند t)، غیر مستقیم (مانند @t) یا مقدار صریح (مانند ♯5) می‌توانید در کد میانی استفاده کنید. توجه کنید که در خروجی نهایی باید به جای t، نشانی مکان این متغیر در حافظه قرار گیرد.
- برای سادگی فرض کنید آدرس متغیرها به صورت ایستا [۴] تخصیص می‌یابد.
- میزان فضای در نظر گرفته شده برای هریک از متغیرها (مانند t) ۴ بایت است.

---

[۳] Addressing Modes
[۴] static

خروجی‌های این فاز و نمونه‌ی ورودی

ورودی شما در این فاز همانند فازهای قبلی، یک فایل متن حاوی کدی است که باید کامپایل کنید. شما باید دو فایل خروجی بدهید که در یکی از آن‌ها، فهرست خطاهای موجود در برنامه به همراه نوع آن‌ها چاپ می‌شود (فرمت دقیق این فایل بر عهده‌ی خودتان است فقط نوع خطا و شماره‌ی خطی که خطا در آن مشاهده شده باید حتما چاپ شود) و در دیگری، کد سه آدرسه‌ی حاصل از کامپایل این کد چاپ می‌شود. دقت کنید کد خروجی شما باید با مفسری که در کنار این سند در اختیارتان قرار گرفته است، قابل اجرا باشد. همچنین در نظر داشته باشید که در مواردی که فایل ورودی حاوی خطا باشد، فایل کد نهایی بررسی نمی‌شود و تنها فهرست خطاهای تشخیص داده شده بررسی می‌شود. در ادامه یک کد نمونه‌ی ورودی به همراه خروجی نهایی آن قرار داده شده است:

```
1   void main(void){
2     int prod;
3     int i;
4       prod = 1;
5       i = 1;
6     while(i < 7){
7       prod = i * prod;
8       i = i + 2;
9     }
10    output(prod);
11    return;
12  }
```

نمونه فایل ورودی

```
0 (ASSIGN,#1,500,)
1 (ASSIGN,#1,504,)
2 (ASSIGN,#7,1000)
3 (LT,504,1000,1004)
5 (JPF,1004,12,)
6 (MULT,500,504,1008)
7 (ASSIGN,1008,500,)
8 (ASSIGN,#2,1016)
9 (ADD,1016,504,1020)
10 (ASSIGN,1020,504,)
11 (JP,3,,)
12 (PRINT,500,,)
```

نمونه فایل خروجی

انجام پروژه

- برای پیاده‌سازی این فاز، در صورتی که گروه‌هایی با توجه به حذف یکی از دانشجویان تک نفره می‌شوند، می‌توانند با اطلاع‌رسانی به دستیاران، اقدام به تشکیل گروه‌های جدید بکنند. سعی کنید در انجام پروژه به صورت گروهی، هر دو عضو گروه، مشارکت کافی داشته باشند. در صورت اختلاف قابل توجه مشارکت دو عضو گروه در انجام پروژه، فردی که مشارکت کمتری داشته، نمره‌ی کمتری نسبت به دیگری می‌گیرد.
- در صورتی که هر گونه سوالی در رابطه با تعریف پروژه دارید، آن را از طریق کوئرا مطرح نمایید.
- مهلت بارگذاری سورس کد پروژه برای فاز آخر، ساعت ۸:۰۰ روز یکشنبه ۹ تیر است.
- زمان‌بندی تحویل حضوری فاز نهایی متعاقبا اعلام خواهد شد. (توجه کنید که حضور هر دو عضو گروه‌های دو نفره در جلسه‌ی تحویل الزامی است.)

موفق باشید.