به نام خدا

اصول طراحی کامپایلر - شرح پروژه (قسمت اول)
دانشگاه اصفهان

مقدمه

در این پروژه قصد داریم برای یک زبان برنامه‌نویسی به نام PL یک کامپایلر طراحی و پیاده‌سازی کنیم. زبان PL یک زبان دستوری شبیه به زبان C است. یک کامپایلر از تحلیل‌گر لغوی، تحلیل‌گر نحوی، تحلیل‌گر معنایی، تولیدکننده و بهینه‌ساز کد میانی، و تولیدکننده و بهینه‌ساز کد اسمبلی تشکیل شده است. یک کامپایلر کامل، یک برنامه نوشته شده در یک زبان برنامه‌نویسی را از یک فایل متنی دریافت کرده، معادل کد اسمبلی آن را تولید می‌کند. کامپایلر طراحی شده در این پروژه یک کامپایلر کامل نیست، بدین معنا که این کامپایلر کد اسمبلی تولید نمی‌کند. کامپایلر طراحی و پیاده‌سازی شده در این پروژه، یک برنامه به زبان برنامه نویسی PL را دریافت کرده، توسط تحلیل‌گر لغوی توکن‌های آن را استخراج می‌کند، و توکن‌های استخراج شده را به تحلیل‌گر نحوی ارسال می‌کند. تحلیل‌گر نحوی، بر اساس گرامر زبان، و یک یا دو الگوریتم تجزیه (مانند تجزیه بالا به پایین و پایین به بالا)، یک درخت نحوی تولید کرده و در صورت وجود خطا، پیام خطا صادر می‌کند. در صورتی که در برنامه ورودی خطایی وجود نداشت، کامپایلر برنامه را پس از تحلیل معنایی اجرا می‌کند.

پیاده‌سازی کامپایلر توسط هر زبانی (مانند سی++، جاوا، و پایتون) می‌تواند انجام شود. تهیه یک گزارش جامع برای توضیح نحوهٔ طراحی هر یک از قسمت‌های کامپایلر الزامی است. در طراحی و پیاده‌سازی کامپایلر دانشجویان می‌توانند از هر منبعی استفاده کنند، اما باید منبع را حتما ذکر کنند. در صورت استفاده از موتورهای هوش مصنوعی، لازم است منبعی که موتور هوش مصنوعی استفاده کرده است ذکر شود. برای مثال بینگ، منابعی را که از آن اطلاعات استخراج می‌کند ذکر می‌کند.

۱ تحلیل‌گر لغوی

در این قسمت از پروژه می‌خواهیم یک تحلیل‌گر لغوی برای زبان PL طراحی و پیاده‌سازی کنیم. تحلیل‌گر لغوی یک فایل حاوی یک برنامه را دریافت می‌کند. در صورتی که برنامه حاوی توکن‌های معتبر در زبان PL باشد، دنباله‌ای از توکن‌های تشخیص داده شده چاپ می‌شوند، و در غیر اینصورت پیام خطا چاپ می‌شود. واژه‌های زبان حساس به حروف کوچک و بزرگ [1] هستند، بنابراین X و x دو کلمه متفات‌اند.

۱.۱ کلمات کلیدی

کلمات کلیدی زبان با حروف کوچک نوشته می‌شوند که در زیر ذکر شده‌اند.

```
bool   break   char   continue   else     false
for    if      int    print      return   true
```

۲.۱ شناسه‌ها

یک شناسه نامی برای یک موجودیت در یک زبان برنامه‌نویسی است. دو موجودیت در این زبان برنامه‌نویسی عبارتند از متغیر و تابع. یک متغیر موجودیتی است که یک یا دنباله‌ای از خانه‌های حافظه را اشغال می‌کند و دارای یک نام است. یک تابع موجودیتی است که تعدادی ورودی دریافت می‌کند، محاسباتی را انجام می‌دهد و در برخی موارد یک مقدار بازمی‌گرداند. یک تابع با یک نام مشخص می‌شود.

شناسه‌ها با یک حرف یا علامت زیرخط [2] _ آغاز می‌شوند و می‌توانند حاوی ارقام، حروف و علامت‌های زیرخط باشند. شناسه‌ها نمی‌توانند برابر با هیچ‌یک از کلمات کلیدی باشند.

۳.۱ علامت‌های نشانه‌گذاری

علامت‌های نشانه‌گذاری در این زبان به شرح زیر هستند.

[1] case-sensitive
[2] underline

- علامت‌های آکولاد باز } و آکولاد بسته { در تعریف بلوک‌ها استفاده می‌شوند.
- علامت‌های پرانتز باز) و پرانتز بسته (در تعریف توابع، فراخوانی توابع، و عبارت‌های محاسباتی استفاده می‌شوند.
- علامت‌های کروشه باز] و کروشه بسته [برای تعریف آرایه‌های استفاده می‌شوند.
- علامت ویرگول , برای جدا کردن ورودی‌های تابع از یکدیگر در تعریف و فراخوانی تابع استفاده می‌شود.
- علامت نقطه‌ویرگول ; در پایان تعریف متغیرها، دستورات محاسبه‌ای و فراخوانی توابع، و همچنین در تعریف حلقه‌ها استفاده می‌شوند.

۴.۱ توضیحات

توضیحات [3] با دو علامت اسلش [4] یا خط اریب // آغاز می‌شوند و با کاراکتر پایان خط معادل کد اسکی ۱۰ یا \n پایان می‌یابند. تحلیل‌گر لغوی توضیحات را به تحلیل‌گر نحوی ارسال نمی‌کند، اما پس از تحلیل لغات لیست همهٔ توکن‌ها را چاپ می‌کند.

۵.۱ مقادیر عددی

مقادیر عددی می‌توانند در مبنای ده (دهدهی یا دسیمال) [5] یا در مبنای شانزده (شانزده‌شانزدهی یا هگزادسیمال) [6] باشند. یک عدد دهدهی می‌تواند مثبت یا منفی باشد که در صورت منفی بودن با علامت - آغاز می‌شوند. اعداد هگزادسیمال با دو کاراکتر 0x آغاز می‌شوند.

۶.۱ کاراکترها و رشته‌های ثابت

یک کاراکتر ثابت، یکی از حروف الفبای اسکی [7] است. یک کاراکتر ثابت بین دو علامت آپوستروف ' ' قرار می‌گیرد. برای نشان دادن علامت آپوستروف در یک کاراکتر ثابت از '\'' و برای نشان دادن کاراکتر خط اریب وارون (بک‌اسلش) [8] از '\\' استفاده می‌شود.

یک رشتهٔ ثابت را با دنباله‌ای از کاراکترها که در بین دو علامت نقل قول " " قرار گرفته‌اند، نشان می‌دهیم. برای نشان دادن علامت نقل قول در یک رشته ثابت از "\ استفاده می‌شود.

۷.۱ عملگرها

در یک عبارت می‌توان از عملگرهای حسابی [9] مانند + ، - ، * ، / برای جمع، تفریق، ضرب، و تقسیم، استفاده کرد. برای به دست آوردن باقیمانده از عملگر % استفاده می‌شود. عملگرهای یگانی + و - برای تعیین مثبت و منفی بودن اعداد به کار می‌روند.

عملگرهای یگانی + و - بالاترین اولویت را دارند و پس از آنها * ، / ، % هم اولویت بوده و در درجهٔ دوم اولویت قرار دارند و در نهایت + و - اولویت سوم قرار می‌گیرند.

عملگرهای رابطه‌ای [10] < ، =< ، > ، => ، == و =! برای مقایسه دو مقدار به کار می‌روند. عملگر < مقدار درست را بازمی‌گرداند اگر عملوند اول از عملوند دوم بزرگتر باشد. به همین ترتیب عملگرهای بعدی مقدار درست را بازمی‌گردانند اگر عملوند اول آنها از عملوند دوم بزرگتر یا مساوی، کوچکتر، کوچکتر یا مساوی، مساوی، نامساوی باشد. عملگرهای رابطه‌ای نسبت به عملگرهای حسابی اولویت کمتری دارند.

عملگرهای منطقی [11] عطف && و فصل || و نقیض ! نیز در عبارات منطقی به کار می‌روند. یک عبارت منطقی عبارتی است که از متغیرهای منطقی و عملگرهای منطقی تشکیل شده و مقدار آن درست یا نادرست است. اولویت عملگرهای منطقی از عملگرهای حسابی و رابطه‌ای کمتر است.

عملگر انتساب = برای مقداردهی یک متغیر به کار می‌رود و اولویت آن از عملگرهای حسابی، مقایسه‌ای، و منطقی کمتر است.

---

[3] comments
[4] slash
[5] decimal
[6] hexadecimal
[7] American Standard Code for Information Interchange (ASCII)
[8] backslash
[9] arithmetic operators
[10] relational operators
[11] logical operator

۸.۱ فاصله‌های خالی

توکن‌ها توسط یک فاصلهٔ خالی [12] و یا ترکیبی از فاصله‌های خالی از یکدیگر جدا می‌شوند. یک فاصله خالی شامل کاراکتر فاصله با کد اسکی ۳۲ ، کاراکتر خط جدید با کد اسکی ۱۰ ، و کاراکتر ستون جدید با کد اسکی ۹ می‌شود.

۹.۱ لیست توکن‌ها

در جدول زیر توکن‌های زبان PL به همراه نام توکن‌ها ذکر شده اند. خروجی تحلیل‌گر لغوی دنباله‌ای از نام توکن‌های یک برنامهٔ ورودی است.

[12] whitespace

Lexeme	Token
`bool`	T_Bool
`break`	T_Break
`char`	T_Char
`continue`	T_Continue
`else`	T_Else
`false`	T_False
`for`	T_For
`if`	T_If
`int`	T_Int
`print`	T_Print
`return`	T_Return
`true`	T_True
`+`	T_AOp_PL
`-`	T_AOp_MN
`*`	T_AOp_ML
`/`	T_AOp_DV
`%`	T_AOp_RM
`<`	T_ROp_L
`>`	T_ROp_G
`<=`	T_ROp_LE
`>=`	T_ROp_GE
`!=`	T_ROp_NE
`==`	T_ROp_E
`&&`	T_LOp_AND
`||`	T_LOp_OR
`!`	T_LOp_NOT
`=`	T_Assign
`(`	T_LP
`)`	T_RP
`{`	T_LC
`}`	T_RC
`[`	T_LB
`]`	T_RB
`;`	T_Semicolon
`,`	T_Comma
variable or function names	T_Id
decimal integers	T_Decimal
hexadecimal integers	T_Hexadecimal
constant strings `"[string]"`	T_String
constant characters `'[character]'`	T_Char
`//[string]\n`	T_Comment
whitespace (newline، tab، and space characters)	T_Whitespace