به نام خدا

اصول طراحی کامپایلر - شرح پروژه (قسمت دوم)
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.

	Lexeme	Token		Lexeme	Token
۱	`bool`	T_Bool	۲۲	`!=`	T_ROp_NE
۲	`break`	T_Break	۲۳	`==`	T_ROp_E
۳	`char`	T_Char	۲۴	`&&`	T_LOp_AND
۴	`continue`	T_Continue	۲۵	`||`	T_LOp_OR
۵	`else`	T_Else	۲۶	`!`	T_LOp_NOT
۶	`false`	T_False	۲۷	`=`	T_Assign
۷	`for`	T_For	۲۸	`(`	T_LP
۸	`if`	T_If	۲۹	`)`	T_RP
۹	`int`	T_Int	۳۰	`{`	T_LC
۱۰	`print`	T_Print	۳۱	`}`	T_RC
۱۱	`return`	T_Return	۳۲	`[`	T_LB
۱۲	`true`	T_True	۳۳	`]`	T_RB
۱۳	`+`	T_AOp_PL	۳۴	`;`	T_Semicolon
۱۴	`-`	T_AOp_MN	۳۵	`,`	T_Comma
۱۵	`*`	T_AOp_ML	۳۶	variable or function names	T_Id
۱۶	`/`	T_AOp_DV	۳۷	decimal integers	T_Decimal
۱۷	`%`	T_AOp_RM	۳۸	hexadecimal integers	T_Hexadecimal
۱۸	`<`	T_ROp_L	۳۹	constant strings `"[string]"`	T_String
۱۹	`>`	T_ROp_G	۴۰	constant characters `'[character]'`	T_Character
۲۰	`<=`	T_ROp_LE	۴۱	`//[string]\n`	T_Comment
۲۱	`>=`	T_ROp_GE	۴۲	whitespace (newline، tab، and space characters)	T_Whitespace

[12] whitespace

۲ تحلیل‌گر نحوی

در این قسمت از پروژه می‌خواهیم یک تحلیل‌گر نحوی برای زبان PL طراحی و پیاده‌سازی کنیم. تحلیل‌گر نحوی دنباله‌ای از توکن‌ها را، که توسط تحلیل‌گر لغوی از یک فایل حاوی برنامه‌ای به زبان PL استخراج شده‌اند، دریافت می‌کند. در صورتی که برنامه فاقد خطای نحوی باشد، درخت نحوی را در خروجی چاپ می‌کند، در غیر اینصورت خطاهای برنامه را گزارش می‌کند. ساده‌ترین روش برای پیاده‌سازی تحلیل‌گر نحوی استفاده از تجزیه‌کنندهٔ پیش‌بینی کننده است. در صورتی که ورودی با خطا روبرو شد، کامپایلر باید با استفاده از روش‌هایی بازیابی خطا انجام دهد.

۱.۲ متغیرها و نوع‌های داده‌ای

سه نوع دادهٔ اصلی در این زبان وجود دارند. نوع داده‌های صحیح، کاراکتر، و منطقی که با `int` و `char` و `bool` نمایش داده می‌شوند. همچنین می‌توان توسط عملگر براکت باز و بسته آرایه تعریف کرد. یک آرایه متغیری است که دنباله‌ای محدود از خانه‌های حافظه را توسط یک نام معین توصیف می‌کند. اندازهٔ آرایه با یک عدد صحیح ثابت تعیین می‌شود. برای تعریف یک متغیر، نوع متغیر و نام آن مشخص می‌شوند. متغیرهایی که هم‌نوع هستند می‌توانند توسط عملگر کاما از یکدیگر جدا شوند. همچنین متغیرها می‌توانند در هنگام تعریف با استفاده از عملگر انتساب مقداردهی شوند. در زیر چند نمونه تعریف متغیر نشان داده شده است.

```
int x;
bool b;
int array[20];
bool booleans[10];
int i=5,j,k=10;
bool t=true, f=false;
char c1, c2='p';
char string[50], s[12] = "PL Compiler";
```

۲.۲ دستورات شرطی

یک دستور شرطی برای بررسی یک شرط و اجرای پاره‌ای از دستورات در صورت برقراری شرط استفاده می‌شود. در یک دستور شرطی از کلمات کلیدی `if` و `else` استفاده می‌شود. در صورتی که شرط متعلق به `if` برقرار باشد، دستورات متعلق به آن که در بلوک متعلق به `if` با استفاده از علامت‌های آکولاد باز و بسته مشخص شده‌اند، اجرا می‌شوند. در صورتی که شرط برقرار نباشد، دستورات متعلق به بلوک `if` اجرا نمی‌شوند. در صورتی که بخواهیم هنگامی که شرط برقرار نیست، دستوراتی را اجرا کنیم می‌توانیم از کلیدواژهٔ `else` و بلوک متعلق به آن استفاده کنیم. امکان تعریف `if` و `else` های تودرتو نیز وجود دارد، بدین معنی که شرطی بررسی شده، در صورت برقراری شرط دستوراتی اجرا می‌شوند و در صورتی که شرط برقرار نباشد و شرط دیگری برقرار باشد، پاره‌ای دیگر از دستورات اجرا می‌شوند. در زیر چند نمونه دستورات شرطی تعریف شده‌اند.

```
if (x==1) {
    y = 6;
    z = 7;
}
if (y<=2 && z>7 || w!=9) {
   x=4+y;
} else {
   x=3*z;
}
if (x>=5 || z<7 && !b) {
    b = true;
} else if (x==3) {
    array[4] = 43;
} else {
    c = 'q';
```

```
}
```

۳.۲ حلقه‌ها

یک حلقه برای تکرار تعدادی دستورات استفاده می‌شود. در یک حلقه از کلمهٔ کلیدی for استفاده می‌شود. حلقه شامل سه بخش مقداردهی اولیه، شرط حلقه، و گام حلقه می‌شود. در مقداردهی اولیه یک متغیر مقداردهی می‌شود، در شرط حلقه یک شرط بررسی شده، و در صورت برقراری شرط، حلقه دوباره تکرار می‌شود، و در گام شرط مقدار متغیر حلقه تغییر می‌کند. این سه قسمت با عملگر نقطه ویرگول از یکدیگر جدا می‌شوند. درون حلقه می‌توان از دستورات break و continue استفاده کرد. اگر از دستور break در یک حلقه استفاده شود، تکرار حلقه متوقف می‌شود و کنترل برنامه از حلقه خارج می‌شود و از اگر دستور continue استفاده شود، دستورات بعد از آن اجرا نمی‌شوند و کنترل برنامه به ابتدای حلقه باز می‌گردد. در زیر چند نمونه حلقه تعریف شده اند.

```
for (int i = 0 ; i<10; i=i+1) {
    array[i] = i;
}
for (x = 100 ; x>0; x=x-2) {
    sum = sum+x;
}
for (j = 0 ; ; j=j+1) {
    sum=sum+j;
    if (j>10) {
        break;
    }
}
for (int i = 0 ; i<100; i=i+1) {
    if (i % 7 == 0) {
        continue;
    }
    s = s+i;
}
```

۴.۲ توابع

یک تابع با تعیین نوع مقدار بازگردانده شده توسط تابع، نام تابع، و پارامترهای ورودی تابع و نوع آنها تعریف می‌شود. یک تابع شامل دسته‌ای از دستورات است که در بلوک تابع بین دو علامت آکولاد باز و بسته نوشته می‌شوند. یک تابع نمی‌تواند یک آرایه بازگرداند. دستور return یک مقدار را از تابع باز می‌گرداند. در زیر چند نمونه تابع تعریف شده‌اند.

```
int multiply(int x, int y) {
     int z = x*y;
     return z;
}
int multiply(int x, bool b, char c) {
     int r;
     if (b && c == 'y') {
          r = x*10;
     } else {
          r = x*5;
     }
```

```
    return r;
}
```

۵.۲ دستور چاپ

از دستور چاپ برای چاپ مقادیر عددی، کاراکترها و رشته‌ها استفاده می‌شود. این دستور یک رشته، و صفر یا تعدادی متغیر را دریافت کرده، مقادیر متغیرها را در درون رشته چاپ می‌کند. رشتهٔ دریاف شده، رشته قالب‌بندی نامیده می‌شود. در درون رشته قالب‌بندی تعدادی تعیین کننده نوع استفاده شده‌اند که برای چاپ اعداد و کاراکترها استفاده می‌شوند. تعیین‌کننده‌های نوع می‌توانند %d برای اعداد صحیح یا %c برای کاراکترها یا %s برای آرایه‌ای رشته‌ها (آرایه‌ای از کاراکترها) باشند. در زیر چند نمونه دستور چاپ نشان داده شده است.

```
print("PL Programming Language");
print("%d", i*j);
print("The value of x is %d and the value of c1 is %c .", x, c1);
print("%s\n", string);
```