پروژه‌ی درس طراحی کامپایلر — معرفی زبان تسلنک

مستند حاضر زبان ساده‌ی تسلنک یا TSLNCC[1] را معرفی می‌کند. در گام‌های تمرین عملی درس کامپایلر، دانشجویان باید برای این زبان یک مفسر یا مترجم بنویسند.

انواع داده

زبان تسلنک دارای دو نوع داده‌ی اصلی می‌باشد:

۱) متغیرهای نوع word یک مقدار عددی علامت‌دار را در خود نگه می‌دارند.

۲) آرایه‌های word که تعدادی متغیر از نوع word را در خانه‌های متوالی حافظه ذخیره می‌سازند.

تعریف متغیرهایی با این دو نوع به شکل زیر انجام می‌گردد:

```
word w;                 # w is a variable of type word
word A[100];            # A is a word array with 100 entries
```

ساختار برنامه‌های تسلنک

برنامه‌های این زبان در یک فایل نوشته می‌شوند که شامل تعدادی تابع می‌باشد (بنابراین در این زبان متغیرهای جهانی[2] وجود ندارند). خط اول هر تابع مشابه زبان C نوع خروجی آن (که تنها word می‌تواند باشد)، نام تابع و پارامترهای آن را مشخص می‌کند. هر یک از پارامترهای تابع می‌تواند از نوع word یا آرایه‌ای از word باشد. بدنه‌ی هر تابع در بین علامت } و { قرار می‌گیرد و شامل تعدادی عبارت یا Statement می‌باشد. شباهت زیادی بین ساختار عبارت‌ها و اولویت عملگرها در زبان تسلنک و زبان C وجود دارد. هر بلاک[3] در این زبان بین دو علامت } و { قرار می‌گیرد. در هر بلاک می‌توان متغیر تعریف نمود و بلاک‌ها می‌توانند تو در تو[4] باشند. حوزه‌ی[5] هر متغیر مشابه زبان C تعریف می‌گردد.

مقدار خروجی یک تابع با استفاده از کلمه‌ی کلیدی return مشخص می‌شود و با اجرای عبارتی که با این کلمه شروع می‌شود، اجرای تابع خاتمه می‌یابد. مثالی از تعریف یک تابع در ادامه نشان داده

---

۱ The Simple Language of NIT Compiler Course
۲ Global variables
۳ Block
۴ Nested
۵ Scope

می‌شود.

```
word sum(word a, word b)
{
	word sum;
	sum = a + b;
	return sum;
}
```

همان طور که اشاره شد، هر تابع می‌تواند آرایه‌ای را نیز به عنوان پارامتر دریافت کند؛ در این صورت، تغییر مقادیر درایه‌ها این آرایه، آرایه‌ی فرستاده شده به تابع را تغییر می‌دهد.

```
word sum100(word A[100])
{
	word sum;
	word i;
	i = 0;
	sum = 0;
	while (i < 100) {
		sum = sum + A[i];
		i = i + 1;
	}
	return sum;
}
```

هر برنامه‌ی تسلنک باید شامل یک تابع با نام main باشد که اجرای برنامه با فراخوانی این تابع آغاز می‌گردد. این تابع بدون پارامتر است و یک word بر می‌گرداند که کد برگشتی برنامه را مشخص می‌نماید.

```
word main()
{
	return 0;
}
```

حلقه‌ها و عبارت‌های شرطی در تسلنک

در زبان تسلنک از عبارت شرطی if و حلقه‌ی while می‌توان استفاده کرد. مثال زیر استفاده از if را نمایش می‌دهد.

```
# inefficient calculation of Fibonacci sequence
word fib(word n)
{
        if (n < 2)
                return 1;
        return fib(n - 1) + fib(n - 2);
}
```

توابع کتابخانه‌ای تسلنک

در تسلنک دو تابع کتابخانه‌ای برای خواندن ورودی و چاپ خروجی وجود دارد:

مقدار یک عدد را از ورودی استاندارد می‌خواند: readword()

مقدار یک عدد را در خروجی استاندارد چاپ می‌نماید: printword()

قواعد تجزیه‌ی زبان تسلنک

در ادامه ساختار BNF زبان تسلنک نمایش داده شده است. در این ساختار اولویت‌های عملگرها (که مشابه عملگرهای زبان C هستند) در نظر گرفته نشده است. همچنین در برنامه‌های زبان تسلنک، علامت # و حروفی که بعد از آن آمده‌اند تا آخر خط comment محسوب می‌شوند.

```
prog ::=    func |
            func prog
func ::=    word iden ( flist ) { body }
body ::=    stmt |
            stmt body
stmt ::=    expr ; |
            def ; |
            if ( expr ) stmt |
            if ( expr ) stmt else stmt |
            while ( expr ) stmt |
            return expr ; |
            { body }
```

```
def ::=    word iden |
           word iden [ num ]
expr ::=   iden = expr |
           iden ( clist ) |
           iden [ num ] |
           expr + expr |
           expr - expr |
           expr * expr |
           expr / expr |
           expr % expr |
           expr < expr |
           expr > expr |
           expr == expr |
           expr <= expr |
           expr >= expr |
           expr | expr |
           expr & expr |
           expr ^ expr |
           expr || expr |
           expr && expr |
           ! expr |
           - expr |
           + expr |
           ~ expr |
           ( expr ) |
           iden |
           num
flist ::=   |
           def |
           def , flist
clist ::=   |
           expr |
           expr , clist
num ::=    [0-9]+
iden ::=   [a-zA-Z_][a-zA-Z_0-9]+
```