آشنایی با زبان برنامه نویسی Haskell

محمدرضا حقیری

فهرست مطالب

هاسکل (به انگلیسی Haskell) یک زبان برنامه نویسی تابعی (Functional) است و برای افرادی که سر و کارشان با فرمولها و الگوریتمهای ریاضی است ، بسیار مناسب میباشد. در این متن سعی داریم که کار با این زبان را تا حدودی در سیستم عامل لینوکس آموزش دهیم. همچنین این متن توسط LaTeX و موتور XeTeX تدوین شده است.

۱ برای شروع به چه چیزی نیاز داریم؟

هاسکل برای پردازش روی اعداد ساخته شده است، پس اینجا به مثال HelloWorld هیچ نیازی نیست. اما به هرحال یک زبان برنامه نویسی بوده و به یک ادیتور و یک محیط کامپایلر نیاز دارد. همچنین این زبان دارای Shell مخصوص به خود نیز هست . (درست همانند پایتون) . برای ادیتور، نیاز نیست حتما ادیتوری پیشرفته نصب کنید (یا بخرید). ساده ترین ادیتور ها مانند nano نیز میتوانند کارتان را راه بیندازند. اگر هم قصد داشته باشید تماما در شل هاسکل باشید ، اصلا حاجتی به ویرایشگر نیست! اما اگر میخواهید توابع و کدهای مورد

نظر را همیشه نگهداری کنید، بهتر است از یک ادیتور مثل Geany یا gedit استفاده کنید. کامپایلر هاسکل ghc است. همچنین به شل ghci نیز نیازمندیم. چنانچه همانند نگارنده از اوبونتو یا دبیان استفاده میکنید کد زیر را در ترمینال اجرا کنید :

```
sudo apt-get install ghc ghci haskell
```

البته هر سه بسته فوق الذکر پیش نیاز دیگری بوده و چنانچه یکی از آنها را نصب کنید هم ، باز دیگر بسته ها همراهش نصب خواهند شد. جهت اطلاع از سایر کامپایلرها میتوانید به وبگاه رسمی [1] هاسکل مراجعه کنید.

۲ هاسکل چه ویژگی هایی دارد؟

هاسکل کمی تنبل است ، درست شنیدید! تنبل! میپرسید چرا؟ برای این که این زبان یک تابع که شما نوشته اید را ، تنها زمانی اجرا میکند که فراخوانی شود. در زمان دیگری آن تابع اجرا نمیشود.

۳ چه زبانهای دیگری عملکردی مشابه هاسکل دارند؟

زبانهای زیادی هستند ، مشهور ترین آنها در دنیای لینوکس ، R است. همچنین در دنیای کاربران سایر سیستم عاملها MATLAB نیز از محبوبیت بالایی برخوردار است. شما مختارید که بین این زبانها ، یک یا چند تا را انتخاب کرده و یاد بگیرید اما پیشنهاد من به شما هاسکل است ، زیرا هم ساده است و هم دید ریاضیاتی شما را بالا میبرد!

۴ مقدمه چینی بس است! شروع کنیم

خب شروع میکنیم. برای شروع کافیست عبارت ghci را در ترمینال تایپ کرده تا با چنین صحنه ای روبرو شوید :

```
GHCi, version 7.4.2: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude>
```

عبارت Prelude در واقع Prompt شل ghci است. بیایید کمی امتحان کنیم . ابتدا چند مساله ساده را به کمک این شل حل میکنیم. هم عبارت های ساده ای مانند ۲+۲ و یا سینوس ۴۵ را به برنامه میدهیم

[1] http://haskell.org

```
Prelude> 2+2
4
Prelude> 2+3
5
Prelude> sin(45)
0.8509035245341184
```

نتیجه را دیدید . بله ، دقیقا همان کارهایی که با ماشین حساب مهندسی انجام میدهیم در این زبان هم امکان پذیر است. اما توجه کنید که اینجا پردازنده کوچک ماشین حساب طرف حساب ما نیست ، حتی اگر از ضیف ترین سیستم مدرن هم استفاده کنید حداقل یک پردازنده ۲ گیگاهرتز ۲ هسته ای دارید که از ماشین حسابهای مهندسی معمولی بسیار قوی تر است. اما آیا باید به همین ها بسنده کنیم؟ نه الزامی ندارد. در ادامه با چند دستور در محیط ghci شما را آشنا میکنیم. شاید شما هم روزانه در ترمینال لینوکس یا مکینتاش یا هر سیستم عامل دیگری مشغول کار میشوید. تبریک! بسیاری از دستورات اینجا نیز شبیه ترمینال یونیکس است. برای تغییر به یک دایرکتوری خاص (مثلا میزکار) باید این دستور را وارد کنید :

```
:cd ~/Desktop
```

همچنین برای خروج از کامپایلر کد زیر را وارد نمایید :

```
:quit
```

اگر خیلی مشتاقید که در مورد دستورات خط فرمان بدانید ، بهتر است به وبسایت دستورات خط فرمان [۲] مراجعه کرده و از آموزشهایش استفاده کنید. همچنین میتوانید به وبلاگ آموزش بش [۳] نیز مراجعه کنید.

۵ تعریف متغیر

همانند سایر محیطها (مثل MATLAB) این محیط هم متغیرهایی در خود دارد. مثلا یکی از معروف ترینها متغیر pi است که همان عدد نام آشنای «پی» است. جهت دیدن مقدار این عدد در شل وارد کنید و نتیجه ای مشابه زیر بگیرید :

```
Prelude> pi
3.141592653589793
```

خب آیا خودمان هم میتوانیم متغیر معرفی کنیم؟ جواب شما بله است. اصل برنامه نویسی و محاسبات در رایانه ، این است که همه چیز را خودمان تعیین کنیم و فقط هنگام اجرای برنامه ها ، مقادیر دلخواه را بعنوان مقدار متغیر ها به آن بدهیم. متغیرها در هاسکل بسیار ساده تعریف میشوند، برای مثال میخواهیم برای r مقدار ۳ را تعیین کنیم. پس به این روش عمل میکنیم :

[۲] http://۳۰li.ir
[۳] http://bash.blogsky.com

```
Prelude> let r = 3
Prelude> r
3
```

توضیح آنکه در خط اول به ازای r مقدار ۳ را تعریف نموده و در خط دوم ، آنرا فراخوانده ایم و سپس ghci نیز به ما نتیجه را نشان داده است. این یک مقدمه است. مقدمه ای برای اینکه در یک فایل دیگر، توابعی را بنویسیم و سپس آن را کامپایل کنیم. به نظرتان چیزی کم نیست؟ درست است تا اینجا در مورد توابع صحبت نکرده ایم. در بخش بعدی به خوبی تکلیف توابع را روشن خواهیم کرد.

۶ توابع در Haskell

همانگونه که میدانید هاسکل یک زبان تابعی است و مطمئنا همین توابع هستند که رنگ و بوی دیگری به هاسکل میدهند. برای اینکه ببینیم توابع در هاسکل چه تعریفی دارند باید تعریف ریاضی تابع را بدانیم. به عبارت دقیقتر هر کجا و در هر زبانی سخن از تابع به میان می آید ، همان تعریف توابع ریاضی را باید دانست. حال ببینیم تعریف ریاضی تابع چیست؟ ریاضی میگوید «تابع قانونی است از A به B که در آن ، به هر عضو A دقیقا یک عضو از B نظیر میشود». مجموعه های A و B هرچیزی میتوانند باشند ، مثلا مجموعه A میتواند افراد یک جامعه و B نیز مجموعه گروه خونی آنان باشد. اما اینجا سر و کار ما با اعداد است. اعداد هستند که در هاسکل میتوانند به عنوان مقدار یک متغیر تعریف شوند. و از همه مهم تر ، هاسکل هم فقط از شما عدد را قبول میکند نه یک سری حروف به عنوان گروه خونی. این همه مقدمه چینی کردیم، حال یک تابع مینویسیم. ما در اینجا یک تابع داریم که به ازای هر ورودی، مساحت دایره ای با شعاع ورودی را میدهد. پس مجموعه A مجموعه تعدادی شعاع و مجموعه B مجموعه مساحتهاست. خب کد و نتیجه کار ما این خواهد شد :

```
Prelude> let area r = pi*(r^2)
Prelude> area 5
78.53981633974483
```

توضیح این کد که واضح است. در خط اول یک تابع نوشتیم. در تابع گفته شد که هر مقداری به ازای r دریافت نمودی ، آن را به توان ۲ برسان و در pi ضرب کن. در خط دوم هم که یک مقدار به ازای r داده ایم ، این عملیات روی آن انجام شد و سپس خروجی عددی گرفتیم. نکته دیگر آنکه دقت هاسکل بسیار بالاست. خب چطور است به همراه هم یک تابع دیگر نیز بنویسیم؟ حال که تا اینجا را خوانده اید ادامه اش را نیز با ما بخوانید. مطمئنا خواندن یک تابع در زبان برنامه نویسی از خواندن درسهایتان جذاب تر است (نگارنده در حال خنده!). یک حقیقت نه چندان تلخ این است که هاسکل نمیتواند کتانژانت حساب کند. خب اینجا به یک حقیقت میرسیم «وقتی یک واحد زنده به محیط بیرون دسترسی ندارد، محیط بیرون به سمت آن می آید»[۴] تابع کتانژانت بسیار ساده است. هاسکل هم سینوس و کسینوس را محاسبه میکند و هم تانژانت را. بخاطر رابطه ساده ای که بین تانژانت و کتانژانت وجود دارد ، تابع را با استفاده از تانژانت مینویسیم :

[۴] زیست اول دبیرستان - فصل دوم

```
Prelude> let cot(r) = 1/tan(r)
Prelude> cot(45)
0.6173696237835551
```

در خط اول به هاسکل فهماندیم که کتانژانت ، تقسیم عدد یک بر تانژانت است (گرچه میتوان از طریق سینوس و کسینوس هم این محاسبات را انجام داد). البته میتوانید امتحان هم بکنید ، امتحان ضرری ندارد. یک متغیر را در نظر بگیرید (مثلا در اینجا همان r و سپس تانژانت و کتانژانت آن را در هم ضرب کنید (در صورتی تابع ما صحیح است که جواب ۱ باشد)

```
Prelude> tan(30)*cot(30)
1.0
```

بله خوشبختانه کد ما درست بوده و حاصلضرب کتانژانت در تانژانت برابر با یک شده است. در بخش بعد در مورد نوشتن کد و سپس کامپایل آن صحبت خواهیم کرد.

۷ نوشتن و کامپایل برنامه هاسکل

همانگونه که کد C را در یک ادیتور مینویسیم و آن را توسط MinGW یا GCC کامپایل میکنیم ، کد هاسکل را هم میتوانیم. برای نوشتن کد هاسکل (همانگونه که ابتدای مقاله ذکر شد) نیازی نیست تا حتما ادیتور بسیار پیشرفته ای داشته باشیم. بهرحال نگارنده ترجیح میدهد از Leafpad (ادیتور پیشفرض میزکار Xfce) استفاده کند. چون در هاسکل کدها نیازی به رنگی شدن و خطایابی ندارند.

نکته : در کدهایی که مینویسیم نیازی به let برای تعریف تابع و متغیر نداریم.

خب همانطور که خواندید ، نیازی نیست که از let استفاده کنیم. میخواهیم کد ساده ای را با هم نوشته و کامپایل کنیم. ابتدا یک داکیومنت خالی روی دسکتاپ ایجاد کرده ، آن را با ادیتور دلخواه باز میکنیم و سپس کدهای زیر را داخلش مینویسیم :

```
f(x) = x^2
```

و با نام function.hs آنرا ذخیر میکنیم. **توجه کنید که سورسهای هاسکل همه پسوند hs دارند** سپس وارد ترمینال و سپس ghci شده و تایپ میکنیم :

```
Prelude> :cd ~/Desktop/
Prelude> :load function.hs
```

کد اول شما را به فولدر Desktop میبرد و کد دوم برنامه را کامپایل میکند. پس از زدن دستور دوم چنین صحنه ای پیش چشمان شما رخ میدهد :

```
[1 of 1] Compiling Main             ( function.hs, interpreted )
Ok, modules loaded: Main.
*Main>
```

و prompt از Prelude به Main تغییر میکند. Main در حقیقت ماژولی است که برنامه از آن به جهت کامپایل شدن استفاده کرده است. چرا بیکارید؟ کمی به جای x مقدار بگذارید و نتیجه را ببینید :

```
*Main> f(2)
4
*Main> f(3)
9
```

درست دیدید! برنامه با موفقیت کامپایل شد و نتیجه داد! این خود نشان میدهد که کد ما مشکلی نداشته است. خب ، آیا اکنون کار دیگری با هاسکل داریم؟ شاید الان دیگر کاری نداشته باشیم. پس به سادگی تایپ میکنیم :

```
*Main> :quit
```

اکنون از شل هاسکل خارج شده و به شل ترمینال (عموما Bash) بر خواهیم گشت و با چنین چیزی روبرو خواهیم بود :

```
prp-e@prpe-Aspire-5520:~$
```

خب تا اینجا حسابی با هاسکل سر و کله زده اید و حتما لذت تایپ دستورات هاسکل را درک کرده اید.

۸ کلام آخر

آنچه در این مقاله خواندید ، تنها یک معرفی و انجام ساده ترین کارها با هاسکل بود. گرچه در آینده این زبان را ادامه خواهیم داد ، اما بهتر است این کار ها مبتنی بر وب باشد. همچنین میتوانید نظرات خود را در مورد این کتابچه در انجمنهای تخصصی ایران بی اس دی [۵] به ما اعلام کنید. همچنین میتوانید در وبلاگ نگارنده [۶] و یا از طریق ایمیل [۷] با او ارتباط برقرار کنید.

و من الله التوفیق

محمدرضا حقیری - بهار ۹۲

---

[۵] http://iran-bsd.ir

[۶] http://haghiri۷۵.com

[۷] haghiri۷۵@gmail.com