بسمه تعالی

زبان برنامه نویسی جولیا و کاربرد

زمستان ۱۳۹۹

فهرست

چکیده..... ۳

فصل اول..... ۴

۱-۱ مقدمه..... ۴

۱-۲ تاریخچه زبان برنامه نویسی جولیا..... ۵

۱-۳ ویژگی‌های زبان برنامه نویسی جولیا..... ۷

۱-۴ کاربرد زبان برنامه نویسی جولیا..... ۸

۱-۵ محاسبات علمی با جولیا..... ۱۱

فصل دوم..... ۱۳

۲-۱ آشنایی با قواعد زبان برنامه نویسی جولیا..... ۱۳

۲-۲ اعداد گویا و مختلط در زبان برنامه نویسی جولیا..... ۱۶

۲-۲-۱ اعداد مختلط..... ۱۶

۲-۲-۲ اعداد گویا..... ۲۱

فصل سوم..... ۲۶

۳-۱ برنامه نویسی موازی..... ۲۶

۳-۲ اصول برنامه نویسی موازی..... ۲۷

۳-۳ امکانات برنامه نویسی موازی..... ۲۸

۳-۴ نمونه برنامه های نوشته شده با جولیا..... ۲۹

منابع..... ۳۰

چکیده

مسلما یکی از سودمندترین اختراعات بشر تا به امروز کامپیوترها بوده اند. دستگاه هایی که هزاران بار سریع تر از انسان فکر می کنند و سرعت عملشان به طرز غیر قابل باوری بالاست. سرعت و قدرت، امکان انجام خیلی از کارهایی را فراهم کردند که انسان به طور عادی از انجام آن ها عاجز بود. اما یک مشکل وجود داشت: این دستگاه ها به همان اندازه که قوی بودند، کم هوش هم بودند؛ آن ها به طور عادی هیچ عملی انجام نمی دادند مگر این که به صراحت از آن ها درخواست می شد. اما این درخواست چگونه باید صورت می گرفت؟ باید راهی برای گفتگو با آن ها پیدا می شد. و در این زمان بود که متخصصان تصمیم گرفتند زبان های مخصوصی را بوجود بیاورند تابتوانند با کامپیوترها گفتگو کنند. این زبان های مخصوص به اصطلاح زبان های برنامه نویسی کامپیوتر نام گرفتند. به نسل اولیه زبان های برنامه نویسی، زبان های سطح پایین گفته می شد، چرا که به سختی قابل یادگیری و به کارگیری بودند. پیاده سازی اعمالی ابتدایی توسط این زبان ها باعث می شد مدت ها وقت با ارزش برنامه نویسان گرفته شود. اما وضع به همین منوال نماند و با گذشت زمان زبان های جدیدی به وجود آمدند که آموختن آن ها راحت تر بود. طی سه دهه ی اخیر، صنعت کامپیوتر مورد هجوم انواع زبان های برنامه نویسی قرار گرفت. در این میان Julia که یک زبان برنامه نویسی نوپا محسوب می‌شود و آزمایشگاه هوش مصنوعی و علوم رایانه وابسته به دانشگاه MIT آن را ساخته است، یکی از محبوب‌ترین پلتفرم‌ها در این زمینه است و کارشناسان حوزه علوم داده توجه فراوان به آن دارند. در سال ۲۰۱۲ برای نخستین بار یک زبان برنامه نویسی توانست توجه جهان را نسبت به خود جلب کند، این زبان سریع، آسان و دقیق بود که جولیا نام گذاری شد. جولیا در حین سادگی می تواند با بزرگترین رقبای خود همچون پایتون و سی پلاس پلاس رقابت کند. امکان برنامه‌نویسی موازی و چندهسته‌ای و رایاناش ابری و نیز محاسبات توزیع شده در جولیا هست. همچنین امکان فراخوانی مستقیم توابع C و Fortran در جولیا وجود دارد. Python و MATLAB نیز با کتابخانه‌های جانبی در جولیا اجرا می‌شوند.

فصل اول

۱-۱ مقدمه

سیستم های کامپیوتری جدید ، تاثیرات وسیع و رشد یابنده ای بر اکثر فعالیت های بشری داشته و دارند. کامپیوتر امکان داده است تا زمینه های جدیدی از تحقیقات در علوم ایجاد شوند که قبلا، به سبب کمبود داده ها و محدودیت در انجام تحلیل ها و محاسبات عددی، چندان شناخته شده نبودند. کامپیوتر، پیشرفت های تکنولوژی، را تسهیل کرده و به عنوان وسیله ای برای کنترل فرایندهای صنعتی، به گستردگی مورد استفاده قرار می گیرد. جولیا یک زبان برنامه‌نویسی پویا و سطح بالا است که بیشتر برای مقاصد محاسبات حجم بالای علمی مورد استفاده قرار می‌گیرد. البته از جولیا می‌توان به عنوان زبان برنامه‌نویسی همه منظوره نیز استفاده کرد. این زبان یک کامپایلر پیشرفته، اجرای همزمان موازی، دقت عددی و یک کتابخانه تابع گسترده ریاضی را فراهم می کند. زبان برنامه نویسی جولیا یک کتابخانه پایه(که عمدتا در جولیا نوشته شده است) و همچنین کتابخانه های متن باز زبان سی و فورترن را برای جبر خطی، تولید عدد تصادفی، پردازش سیگنال و پردازش رشته با هم ترکیب می کند. علاوه بر این، جامعه توسعه دهنده زبان برنامه نویسی جولیا با تعدادی بسته خارجی از طریق مدیریت بسته های جولیا به سرعت در حال گسترش کتابخانه‌های این زبان هستند. در ابتدا برای پاسخگویی به نیاز های تجزیه و تحلیل عددی با کارایی بالا و علوم محاسباتی طراحی شده بود. از ارزیابی مشتاقانه استفاده می کند و شامل کتابخانه های کارامد برای محاسبات نقطه شناور، جبر خطی و تولید عدد تصادفی می باشد. بسیاری از کتابخانه ها در این زبان در دسترس هستند و می توان از ان ها استفاده کرد (به عنوان مثال کتابخانه برای تبدیل سریه فوریه). این زبان برای اولین بار در سال ۲۰۱۲ به طور عمومی انتشار یافت و مورد استفاده قرار گرفت. مهمترین ویژگی های آن سادگی در نصب، یادگیری و کاربری، سرعت بالا در انجام محاسبات و توانمندی های بروز و گسترده در کاربردهای متنوع آمار، ریاضات و محاسبات مهندسی می باشد. این زبان قابلیت های سطح پایین مانند محاسبات ریاضی پایه بر اعداد (صحیح و اعشاری) و اجرای حلقه های تکرار را با سرعت و دقت

بالا (قابل مقایسه با C یا Fortran) انجام می دهد و توانایی های سطح بالایی مانند عملیات ریاضی پیشرفته بر بردارها و ماتریس ها، پردازش موازی و ارائه سرویس بر روی شبکه را پشتیبانی می کند. نحوه کمپایل آن JIT[1] می باشد. به این معنا که برگرداندن کدها به زبان ماشین در زمان اجرا و توسط LLVM[2] انجام می شود. از جمله ویژگی های مهم این زبان پشتیبانی از عبارات ریاضی[3] به صورت یک شی داخلی است. این زبان برنامه نویسی مورد حمایت دانشگاه MIT می باشد و پروفسور Alan Edelman استاد ریاضی پیشرفته این دانشگاه از مدیران فعال پروژه زبان برنامه نوسی Julia است. بعلاوه یکی از وجوه تمایز زبان برنامه نویسی متن باز[4] بودن می باشد، بنابراین کلیه کدهای کمپایلر از طریق سورس جولیا در اختیار عموم قرار دارد.

۱-۲ تاریخچه زبان برنامه نویسی جولیا

Viral Shahو Jeff Bezanson موسسین شرکت Julia Computing، پروژه خود را را در سال ۲۰۰۹ آغاز کردند که بعداً تبدیل به زبان برنامه‌نویسی جولیا شد. رشته تحصیلی Viral Shah علوم کامپیوتر و پایان‌نامه وی در دانشگاه کالیفرنیا بر روی محاسبات موازی متمرکز بود و بعدتر بخشی از محصول Star-P در شرکت Interactive Supercomputing Corp شد. هنگامی که که مایکروسافت این شرکت را خریداری کرد، Viral Shahو Jeff Bezanson کار مشترک خود را در مورد روش‌های جدید محاسبات موازی آغاز کردند. به طور هم‌زمان Viral Shah و Stefan Karpinski در مورد مشکلات مشابهی که در همکاری تحقیقاتی‌شان در UCSB داشتند، همکاری نموده و پس از آن Jeff Bezanson به Alan Edelman پیشنهاد پیوستن به ادامه تحصیلات Ph.D در MIT را ارائه کرد و به این صورت این ۴ نفر یک‌جا جمع شدند.

[1] Just in Time
[2] Low-Level Virtual Machine
[3] Expressions
[4] open source

این چهار نفر در سال ۲۰۰۹ موفق شدند مشکل «دو زبانی» را حل کنند. مهم‌ترین چیزی که مانع پیشرفت آن ها در محاسبات موازی می‌شد این حقیقت بود که کاربرانی که از زبآن های برنامه‌نویسی سطح بالایی مانندR و Python استفاده می‌کنند باید بخش‌های اجرایی را در C یا C++ بازنویسی کنند. این مشکل به شدت باعث پایین آمدن کارایی می‌شد چرا که این کار همیشه موجب به وجود آمدن خطاهای انسانی، از بین رفتن وقت و تلاش، کاهش سرعت فروش و در نهایت عقب افتادن از رقبا می‌شد. این مشکل دو زبانی نه تنها در کار محققین، بلکه در کار دانشمندان، داده پژوهان، مهندسین و تحلیلگران مالی موانع زیادی ایجاد می‌کرد. از این جا بود که زبان برنامه نویسی Julia بوجود آمد.

نخستین نسخه این زبان البته در سال ۲۰۱۲ به صورت رسمی منتشر شد. جولیا زبانی است که از سوی آزمایشگاه هوش مصنوعی و علوم رایآن های دانشگاه MIT موسوم به CSAIL توسعه پیدا کرده است. جولیا سینتکس ساده‌ای شبیه پایتون دارد. سرعت اجرای آن به زبان C نزدیک است. همگانشی همچون Lisp دارد. عملیات ریاضی و ماتریسی شبیه MATLAB دارد و خلاصه تجربه‌های مثبت زبآن های مختلف را در خود گرد آورده است. زبانی که متن‌باز بوده، ۱۹۰۰ بسته ثبت شده برای آن عرضه شده و بیش از ۷۰۰ انجمن فعال متشکل از برنامه‌نویسان مجرب از آن پشتیبانی می‌کنند. زبان برنامه‌نویسی جولیا که مدت زمان زیادی از شکل‌گیری آن نمی‌گذرد، در سال ۲۰۱۸ از نظر سرعت رشد تمامی رقبای خود را پشت سر گذاشته است. زبان برنامه‌نویسی جولیا در سال ۲۰۰۹ توسط محققان دانشگاه ام‌آی‌تی ابداع شد و برای اولین بار در سال ۲۰۱۲ در دسترس عموم قرار گرفت. این زمان برنامه‌نویسی منبع باز به سرعت در حال رشد است و در حال تبدیل شدن به یکی از ۱۰ زبان برنامه‌نویسی محبوب دنیا است. اگر چه جولیا هنوز به اندازه زبان برنامه‌نویسی همچون Python محبوب نیست، اما فعلا در فهرست ۵۰ زبان برنامه‌نویسی برتر دنیا جای دارد و صاحب نظران می‌گویند آینده‌ای درخشان در انتظار جولیاست. در سایت Julia Computing شرکتی که توسط چهار خالق زبان برنامه‌نویسی جولیا تاسیس شده، جولیا زبان برنامه‌نویسی توصیف شده که از توانمندی‌های کمی زبآن های R وPython، سرعت تولید و به نتیجه رساندن زبان جاوا و قدرت زبان سی پلاس پلاس برای حل مشکلات مربوط به حجم بالای داده و تحلیل آن ها برخوردار

است. میزان بارگذاری زبان برنامه‌نویسی جاوا از ژانویه سال ۲۰۱۹ به بعد ۷۸ درصد رشد کرده و از ۸/ ۱ میلیون بار به ۲/ ۳ میلیون بار رسیده است. از جولیا برای پردازش انبوهی از داده در ابررایانه‌ها استفاده می‌شود و به‌عنوان مثال از آن برای شبیه‌سازی ۱۸۸ میلیون ستاره و کهکشان در ابررایآن های به نام کوری استفاده شده است. برای این کار از ۶۵۰ هزار هسته پردازشی ابررایانه مذکور استفاده شده است. این زبان برنامه‌نویسی در حوزه‌هایی همچون تولید خودروهای خودران، مسیریابی انواع وسایل نقلیه، تولید روبات‌ها، چاپگرهای سه‌بعدی، ابزار مرتبط با واقعیت مجازی و واقعیت افزوده، هوش مصنوعی و غیره کاربرد دارد. رشد سالانه این زبان ۱۰۱ درصد گزارش شده و تاکنون نزدیک به دو میلیون بار دانلود شده است. همگی این آمارها بیان‌گر این موضوع هستند که ما با یک زبان روبه‌رشد سروکار داریم. در حالی که جولیا هنوز موفق نشده است به فهرست ۱۰تایی برتر زبآن های برنامه‌نویسی وارد شود اما TIOBE و RedMonk که از سایت‌های مشهور در زمینه ارزیابی زبآن های برنامه‌نویسی هستند به این حقیقت اذعان دارند که جایگاه این زبان نزد توسعه‌دهندگان به سرعت در حال افزایش است Aviva, BlackRock, Capital One و نت‌فلیکس در کنار ۷۰۰ موسسه پژوهشی از جمله شرکت‌ها و سازمآن های بزرگی هستند که از این زبان استفاده می‌کنند.

۱-۳ ویژگی‌های زبان برنامه نویسی جولیا

۱- یک زبان برنامه‌نویسی متن باز با پروانه ام‌آی‌تی است.

۲- مانند زبان لیسپ هومویکونیک است، به این معنا که کدهای برنامه نیز داده‌هایی از برنامه به‌شمار می‌روند، که امکان تولید کدهای برنامه‌نویسی پویا را به برنامه‌نویس می‌دهد.

۳- در تعریف توابع بسیار قوی و انعطاف‌پذیر است، که امکان تعریف رَویه‌ها و عملگرهای محاسباتی جامع برای ساختارهای داده متفاوت را فراهم می‌آورد.

۴- دارای قابلیت‌های سطح پایین کنترلی و محاسباتی است، که سرعت اجرا را تا حد زبآن های با نوع داده ایستا مانند سی یا فرترن افزایش می‌دهد.

۵- دارای سیستم مدیریت بسته‌است.

۶- دارای قابلیت فراخوانی توابع پایتون با استفاده از بسته PyCall است.

۷- دارای قابلیت فراخوانی مستقیم توابع سی است.

۸- نوع داده‌های تعریف شده به وسیله کاربر سرعت و فشردگی انواع توکار زبان را دارند.

۹- پشتیبانی بهینه از یونی‌کد شامل UTF-8

۱۰- برای یادگیری و استفاده ساده است.

۱-۴ کاربرد زبان برنامه نویسی جولیا

براساس اطلاعات ارائه‌شده، زبان برنامه‌نویسی جولیا در سال ۲۰۱۲ با هدف داشتن سرعت زبان سی، قابلیت‌های پایتون، پویایی روبی و قدرت بالای متلب در انجام عملیات سنگین ریاضی در کنار ویژگی‌های آر برای انجام عملیات آماری رونمایی شده است.

جولیا قابلیت‌های متنوع و متعددی دارد و همین امر جولیا را به گزینه‌ی مناسبی برای استفاده در کاربردهای گوناگون تبدیل کرده است. برای مثال جولیا این امکان را دارد تا بتوان با استفاده از آن وظایف پردازشی را بین هسته‌های مختلف تقسیم کرد که این ویژگی امکان استفاده از جولیا از یادگیری ماشین تا شبیه‌سازی ابررایانه‌های بزرگ را فراهم می‌کند.

MIT اعلام کرده که زبان برنامه‌نویسی جولیا را باید پویاترین زبان برنامه‌نویسی در باشگاه زبآن های برنامه‌نویسی موسوم به پتافلاپ خواند. باشگاه پتافلاپ به گروهی از زبآن های برنامه‌نویسی اطلاق می‌شود که با استفاده از آن ها می‌توان از قدرت پردازشی یک پتا فلاپ در ثانیه عبور کرد. براساس اطلاعات

ارائه‌شده، محققان با استفاده از جولیا اقدام به شبیه‌سازی ۱۸۸ میلیون ستاره و کهکشان در سوپرکامپیوتر Coriکرده‌اند که دهمین ابررایانه‌ی قدرتمند در جهان است. این شبیه‌سازی در کمتر از ۱۵ دقیقه انجام شده است. در این شبیه‌سازی، بیش از ۶۵۰ هزار هسته‌ی پردازشی Knights Landing Xeon Phi مورد استفاده قرار گرفته، که نتیجه‌ی آن قدرت پردازشی ۱٫۵ پتافلاپس بوده است.

از جمله‌ی دیگر کاربردهای جولیا می‌توان به استفاده از آن در خودروهای خودران و پرینترهای سه بعدی در کنار پزشکی و لوازم پزشکی با دقت بالا، واقعیت افزوده، ژنتیک، یادگیری ماشین و مدیریت ریسک اشاره کرد.

از جمله‌ی سیستم‌های توسعه‌یافته با استفاده از جولیا باید به توسعه‌ی نسل بعدی سیستم جلوگیری از برخورد هواپیما، بهبود مسیریابی اتوبوس مدارس بوستون و همچنین سیستم حرکتی مسیریابی ربات اشاره کرد. بهره‌مندی از پارادیوم‌های محاسباتی همچون برنامه‌نویسی تابعی و شی‌گرایی، اسکریپت‌نویسی روان، پشتیبانی از برنامه‌نویسی اعلانی، ارائه یک ترکیب نحوی خوب مناسب برای عملیات ریاضی سنگین، ارائه انواع مختلفی از داده‌های عددی و پشتیبانی داخلی از تکنیک‌های برنامه‌نویسی موازی همگی باعث شده‌اند تا جولیا به یک زبان آینده‌دار تبدیل شود. به طور کلی در بررسی کاربردهای زبان برنامه نویسی جولیا می توان به موارد زیر اشاره کرد:

۱-مالی و سرمایه گذاری

از FinTech تا RegTech، معامله‌گران الگوریتمی، مدیران صندوق، مدل‌سازان اقتصاد کلان، تحلیل‌گران و تنظیم کننده‌ها به صورت گسترده‌ای به جولیا روی آورده‌اند.

۲-یادگیری ماشین و هوش مصنوعی

زبان برنامه نویسی جولیا، قدرتی بالا و انعطاف‌پذیری کاملی را برای تمام نیازهای کسانی که در حوزه یادگیری ماشین فعالیت دارند ارائه می‌دهد.

۳- واحدهای پردازش گرافیکی[1]

سرعت فوق العاده، قابلیت‌ها، انعطاف پذیری و ظرفیت‌های جولیا، آن را اولین انتخاب برای GPU و دیگر ابرکامپیوترهایی می‌کند که از شتاب‌دهنده‌ها استفاده می‌کنند.

۴-پردازش موازی

جولیا، ساخت‌های اولیه درونی را هم برای حافظه به اشتراک گذاشته شده و هم برای توزیع‌های موازی ارائه داده است.

۵-علوم زیستی

ما معتقدیم که جولیا فقط زبانی برای آینده نیست بلکه زبانیست که چالش‌های کنونی را نیز با قدرت حل می‌کند. این زبان راه حلی مدرن برای مسائل مدرن است که به سادگی با جدیدترین چالش‌ها سازگار می‌شود. این دلیلی است که ثابت می‌کند چرا ما احساس می‌کنیم که جولیا انتخابی صحیح برای علوم زیستی در صنعت و پژوهش‌ها است.

۶-انرژی

انرژی دارای ویژگی‌های منحصر به فردی است که جولیا را تبدیل به برترین انتخاب برای معامله، پیش‌بینی عرضه و تقاضا، شبکه مدیریت و غیره می‌کند.

۷-بهینه سازی

شناسایی راه حل بهینه بیشتر اوقات نیازمند آنالیز میلیاردها سناریوی ممکن است. زبان برنامه نویسی جولیا طراحی شده تا عدم محدودیت تعریف مدل، تعیین پارامتر و بهینه سازی در یک زبان واحد سریعتر، ساده‌تر و بهتر بتواند محقق شود.

[1] GPU

۸-دولت

سرعت، عملکرد و سهولت استفاده‌ی جولیا باعث شد تا ناسا[1]، بانک فدرال رزرو نیویورک و آزمایشگاه ملی وزارت انرژی ایالات متحده زبان برنامه نویسی جولیا را پذیرفته و استفاده کنند.

۱-۵ محاسبات علمی با جولیا

محاسبات علمی به طور سنتی نیاز به بالاترین عملکرد دارند، با این حال، متخصصان به طور گسترده ای به زبان های پویای سطح پایین برای کار روزانه نقل مکان کرده اند. ما معتقدیم دلایل خوبی برای ترجیح دادن زبان های پویا برای این کاربردها وجود دارد و ما انتظار نداریم استفاده از آن ها کاهش یابد. خوشبختانه، طراحی زبان مدرن و تکنیک های کامپایلر باعث می شود که بیشترین کارایی را به کار ببریم و یک محیط به اندازه کافی واحد برای تولید نمونه اولیه و به اندازه کافی کارآمد برای گسترش برنامه های کاربردی پر کاربرد فراهم کنیم. زبان برنامه نویسی جولیا این نقش را ایفا می کند: این یک زبان پویای انعطاف پذیر است که برای محاسبات علمی و عددی مناسب است و کارایی قابل مقایسه ای با زبان سنتی استاتیک دارد.

از آنجا که کامپایلر جولیا متفاوت از مفسرانی است که برای زبان هایی مانند Python یا R استفاده می شود، ممکن است متوجه شوید که عملکرد جولیا در ابتدا غیرقابل شناخت است.

زبان برنامه نویسی جولیا سهولت و روشنی را برای محاسبات عددی در سطح بالا فراهم می کند، همانند زبان هایی همچون R، MATLAB و Python، اما جولیا همچنان از برنامه های عمومی نیز پشتیبانی می کند. برای رسیدن به این هدف، جولیا در اصل بر مبنای زبان های برنامه نویسی ریاضی متکی است، اما همچنین از زبان های پویای محبوبی از جمله Lisp، Perl، Python، Lua، Ruby ویژگی های زیادی را قرض گرفته است.

[1] NASA

مهمترین خروجی های جولیا از زبان های پویای معمول عبارتند از:

- هسته برنامه نویسی جولیا خیلی کم تحمیل میکند، کتابخانه استاندارد در خود جولیا نوشته شده است، از جمله عملیات اولیه مانندعملگرهای ریاضیات در حساب اعداد صحیح
- زبانی غنی از حروف برای ساخت و توصیف اشیاء است که می توانند به صورت اختیاری برای اعلامیه ها به کار بروند
- توانایی تعریف رفتار تابع از میان ترکیب بسیاری از استدلال ها و استفاده از اعزام های چندگانه
- تولید خودکار کد کارآمد و ویژه برای انواع استدلال مختلف
- عملکرد خوب، نزدیک شدن به زبان های آماری استاتیک مانند C

اگر چه گاهی اوقات از زبان های پویا به عنوان ''typeless'' صحبت می کنند، اما در واقع این طور نیست: در واقع هر شیء، چه اولیه باشد یا از تعاریف کاربر، یک type محسوب می شود. فقدان اعلان های تایپ در اغلب زبان های پویا به این معناست که کامپایلر نمی تواند نوع مقادیر را آموزش دهد و اغلب نمی تواند صراحتا در مورد تایپ ها صحبت کند. از سوی دیگر، در زبان های استاتیک، در حالی که می توانید معمولا نوع حاشیه نویسی برای کامپایلر را داشته باشید، تایپ ها فقط در زمان کامپایل وجود دارند و نمی توانند در زمان اجرا دستکاری یا بیان شوند. در جولیا، تایپ ها خودشان اشیاء زمان-اجرا[1] هستند و همچنین می توانند برای انتقال اطلاعات به کامپایلر استفاده شوند. در حالی که برنامه نویس ممکن است گاه گاه نیازی به استفاده صریح از تایپ ها یا چندین اعلانی نداشته باشد، آن ها ویژگی های اصلی تشکیل جولیا هستند: توابع در ترکیبات مختلف از انواع استدلال تعریف شده و توسط اعزام کردن به مشخص ترین تعریف تطبیق داده شده، به کار گرفته می شوند.

[1] run-time

فصل دوم

۲-۱ آشنایی با قواعد زبان برنامه نویسی جولیا

ساده ترین راه برای یادگیری جولیا، اجرای خود جولیا REPL[1] است. برای اجرای آن یا بر روی آن دابل کلیک کنید و یا از خط فرمان سیستم julia را صدا بزنید. البته برای اجرا در خط فرمان باید جولیا در patch شما موجود باشد.

```
12345julia> 1 + 2
3

julia> ans
3
```

برای خروج از این محیط تعاملی می توانید از ترکیب CTRL-D (گرفتن کلید کنترلسپس فشردن کلید d) یا فرمان ()exit را وارد کنید. همانطور که در محیط نرم افزار می بینید جولیا پس از اجرا شدن در محیط تعاملی یک بنر را نمایش می دهد و در پایین آن منتظر ورودی کاربر می شود. زمانی که کاربر یک دستور کامل را به جولیا بدهد(مثل ۲+۱) سپس کلید enter را فشار دهد، محیط تعاملی آن را بررسی کرده و ارزش آن را نمایش می دهد. اگر در محیط تعاملی پس از دستور خود سمیکالن ; بگذارید و کلید enter را بفشارید، ارزش کد شما (خروجی بیان) نمایش داده نمی شود. متغیر ans ارزش آخرین بیان(دستور) شما را نشان می دهد و فرقی ندارد که قبلا نمایش داده شده یا از نمایش آن بوسیله ; جلوگیری شده است. متغیر ans فقط در محیط تعاملی[2] کار می کند و در صورت های دیگری که کد جولیا را اجرا می کنید کارایی ندارد.

[1] read-eval-print loop
[2] interactive sessions

برای ارزیابی عملکرد یک سورس کد در فایلی به مشخصات file.jl باید include("file.jl") را بنویسید(فرمان دهید). توجه داشته باشید که این فایل باید در پوشه ای که جولیا را نصب کرده این موجود باشد.

اگر می خواهید کد درون فایل را به صورت غیر تعاملی اجرا کنید باید در خط فرمان خود فایل را به عنوان اولین آرگومان(ورودی) دستور julia قرار دهید (توجه کنید که در خط فرمان ویندوز به جای ' باید از " استفاده کنید اما ما تمام دستورات زیر را در لینوکس نوشته‌ایم):

```
1$ julia script.jl arg1 arg2...
```

همانطور که در مثال بالا مشاهده می کنید، خط فرمان نوشته شده آرگومان های بعد از آرگومان فایل را به صورت خط فرمان تفسیر کرده و آن ها را به script.jl می فرستد که به ثابت جهانی ARGS پاس داده می شوند. نام اسکریپت نیز خودش به شاخص جهانی PROGRAM_FILE پاس داده می شود. توجه داشته باشید زمانی که اصطلاح julia در خط فرمان عبارت e- را می گیرد نیز ARGS در کار وجود دارد اما خبری از PROGRAM_FILE نیست. برای آشنایی بیشتر با فرمان های خط فرمان برای جولیا کافیست h- را به جولیا در خط فرمان بدهید. به عنوان مثال، برای فقط چاپ آرگومان داده شده به یک اسکریپت، شما می توانید این کار را انجام دهید:

```
1234$ julia -e 'println(PROGRAM_FILE); for x in ARGS; println(x); end' foo bar

foo
bar
```

یا می توانید کد را درون اسکریپتی قرار داده سپس آن را اجرا کنید:

```
12345$ echo 'println(PROGRAM_FILE); for x in ARGS; println(x); end' > script.jl
$ julia script.jl foo bar
script.jl
foo
```

bar

نماد -- می تواند میان آرگومان های اختصاص داده شده به اسکریپت و آرگومان های اختصاص داده شده به جولیا تفاوت ایجاد کند. نماد -- آرگومان ها را به جولیا می فرستد:

```
1$ julia --color=yes -O -- foo.jl arg1 arg2..
```

جولیا با هر دو دستور انتخابی p- یا machine-file-- می تواند در حالت پردازش موازی (parallel) شروع به کار کند. p n- به اندازه n عملیاتگر پردازش را اجرا می کند در حالی که machine-file file-- به ازای هر خط از فایل file یک عملیاتگر را اجرا می کند. ماشین های تعریف شده برای file باید از طریق بی نیازی رمز(password-less) ssh لاگین[1] شده و در دسترس قرار بگیرند که جولیا نیز باید در همان مکان به عنوان میزبان فعلی نصب شده باشد. هر ماشینی که تعریف می شود فرم زیر را می گیرد:

```
[count*][user@]host[:port] [bind_addr[:port]]
```

پیش فرض user همان کاربر جاری سیستم و port همان پورت استاندارد ssh است. count میزان عملیاتگرهایی است که برروی حل گره کار می کنند که مقدار پیشفرض آن یک است. گزینه bind-to bind_addr[:port] آدرس آی پی و پورتی که دیگر عملیاتگرها برای ارتباط با این عملیاتگر بهتر است استفاده کنند را مشخص می کند.

اگر شما کدی دارید که می خواهید به هنگام اجرای جولیا آن هم اجرا شود، کافیست آن را در ~/.julia/config/startup.jl بگذارید:

```
12345$ echo 'println("Greetings! 你好! 안녕하세요?")' > ~/.julia/config/startup.jl
$ julia
Greetings! 你好! 안녕하세요?
```

...

[1] login

راه های مختلفی برای اجرای کد جولیا (julia) وجود دارد و شبیه گزینه هایی هستند که برای برنامه های پرل[1] و روبی[2] ارائه شده‌اند:

1julia [switches] -- [programfile] [args...]

۲-۲ اعداد گویا و مختلط در زبان برنامه نویسی جولیا

زبان برنامه نویسی جولیا تایپ‌های از پیش تعریف شده‌ای برای هر دوی اعداد گویا و مختلط را در خود دارد و تمام عملیات ریاضیاتی و توابع اولیه مربوط به آن ها را پشتیبانی می‌کند. تبدیل و ارتقاء نیز تعریف شده‌اند که در نتیجه عملیات روی هر ترکیب از تایپ‌های عددی از پیش تعریف شده، خواه تایپ‌های اولیه و خواه ترکیب شده باشند، همانگونه که انتظار می‌رود رفتار می‌کند.

۲-۲-۱ اعداد مختلط

ثابت جهانی im برای عدد مختلط i (که ریشه دوم عدد ۱- است) تخصیص داده شده است. (نماد اختصاصی ریاضی‌دانان i یا مهندسان j را می‌توانید برای این ثابت جهانی به کارگیرید.) از آنجایی که زبان برنامه نویسی جولیا اجازه می‌دهد که لیترال‌های عددی به عنوان ضریب متغیرها بدون فاصله قبل از آن ها قرار گیرند، این قرارگیری به اندازه کافی برای ارائه نحو(سینتکس) مناسب برای اعداد مختلط، شبیه به نماد سنتی ریاضی است:

```
julia> 1+2im
1 + 2im
```

شما می‌توانید تمام عملیات ریاضی استاندارد را بر روی اعداد مختلط انجام دهید:

[1] perl
[2] ruby

```
julia> (1 + 2im)*(2 - 3im)
8 + 1im

julia> (1 + 2im)/(1 - 2im)
-0.6 + 0.8im

julia> (1 + 2im) + (1 - 2im)
2 + 0im

julia> (-3 + 2im) - (5 - 1im)
-8 + 3im

julia> (-1 + 2im)^2
-3 - 4im

julia> (-1 + 2im)^2.5
2.729624464784009 - 6.9606644595719im

julia> (-1 + 2im)^(1 + 1im)
-0.27910381075826657 + 0.08708053414102428im

julia> 3(2 - 5im)
6 - 15im

julia> 3(2 - 5im)^2
-63 - 60im

julia> 3(2 - 5im)^-1.0
0.20689655172413796 + 0.5172413793103449im
```

مکانیسم ارتقاء، اطمینان می‌دهد که ترکیب‌های عملوندهای مختلف دارای تایپ‌های متفاوت، به درستی کار کند:

```
julia> 2(1 - 1im)
2 - 2im

julia> (2 + 3im) - 1
1 + 3im

julia> (1 + 2im) + 0.5
1.5 + 2.0im

julia> (2 + 3im) - 0.5im
2.0 + 2.5im

julia> 0.75(1 + 2im)
0.75 + 1.5im

julia> (2 + 3im) / 2
1.0 + 1.5im

julia> (1 - 3im) / (2 + 2im)
-0.5 - 1.0im

julia> 2im^2
-2 + 0im

julia> 1 + 3/4im
1.0 - 0.75im
```

توجه داشته باشید که 3/4im == 3/(4*im) == -(3/4*im) و این به آن دلیل است که یک ضریب لیترال چسبیده به متغیر، تقدم بیشتری نسبت به تقسیم دارد.

توابع استانداردی جهت مدیریت مقادیر مختلط تدارک دیده شده است:

```
julia> z = 1 + 2im
1 + 2im

julia> real(1 + 2im) # real part of z
1

julia> imag(1 + 2im) # imaginary part of z
2

julia> conj(1 + 2im) # complex conjugate of z
1 - 2im

julia> abs(1 + 2im) # absolute value of z
2.23606797749979

julia> abs2(1 + 2im) # squared absolute value
5

julia> angle(1 + 2im) # phase angle in radians
1.1071487177940904
```

به طور معمول، مقدار قدر مطلق (abs) یک عدد مختلط، فاصله آن را از صفر نشان می‌دهد abs2 . توان دوم مقدار قدر مطلق را می‌دهد و استفاده از آن انکار ناپذیر است چرا که برای کار با اعداد مختلط ریشه دوم بسیار با اهمیت است angle . برای اعداد مختلط فاز زاویه را به رادیان می‌دهد(به عنوان

تابع argument یا arg نیز شناخته می‌شود). محدوده وسیع دیگر توابع ابتدایی، برای اعداد مختلط نیز تعریف شده‌اند:

```
julia> sqrt(1im)
0.7071067811865476 + 0.7071067811865475im

julia> sqrt(1 + 2im)
1.272019649514069 + 0.7861513777574233im

julia> cos(1 + 2im)
2.0327230070196656 - 3.0518977991518im

julia> exp(1 + 2im)
-1.1312043837568135 + 2.4717266720048188im

julia> sinh(1 + 2im)
-0.4890562590412937 + 1.4031192506220405im
```

توجه داشته باشید که معمولاً توابع ریاضیاتی وقتی روی مقداری حقیقی اعمال می‌شوند، عددی حقیقی را برمی‌گردانند و زمانی که روی مقداری مختلط اعمال می‌شوند، عددی مختلط را بر می‌گردانند. برای مثال، sqrt میان اعمال بر روی -1در مقابل (-1) + 0im متفاوت عمل می‌کند اگر چه (-1) + 0im == -1:

```
julia> sqrt(-1)
ERROR: DomainError with -1.0:
sqrt will only return a complex result if called with a complex argument. Try sqrt(Complex(x)).
Stacktrace:
[...]

julia> sqrt(-1 + 0im)
```

```
0.0 + 1.0im
```

نماد گذاری ضریب عددی لیترال برای ساخت عدد مختلط، کارایی ندارد. به جای آن، ضرب باید به صورت صریح بیان شود:

```
julia> a = 1; b = 2; a + b*im
1 + 2im
```

با این حال، این کار توصیه نمی‌شود. به جای آن از تابع مؤثر complex برای ساخت مقدار مختلط استفاده کرده و آن را به طور مستقیم از قسمت‌های حقیقی و موهومی‌اش پدید آورید:

```
julia> a = 1; b = 2; complex(a, b)
1 + 2im
```

این نوع ساخت، از ضرب و عملیات اضافی جلوگیری می‌کند.

Inf و NaN از طریق اعداد مختلط گسترش می‌یابند، در قسمت‌های حقیقی و موهومی یک عدد مختلط که در قسمت مقادیر ویژه اعشاری شرح داده شده:

```
julia> 1 + Inf*im
1.0 + Inf*im

julia> 1 + NaN*im
1.0 + NaN*im
```

۲-۲-۲ اعداد گویا

زبان برنامه نویسی جولیا دارای تایپ عدد گویا است که نسبت‌های دقیق اعداد صحیح را نمایش می‌دهد. اعداد گویا با استفاده از عملگر //ساخته می‌شوند:

```
julia> 2//3
2//3
```

اگر صورت و مخرج یک نسبت دارای فاکتورهای(ضریب‌های) مشترک باشند، با یک دیگر ساده شده و به شکل کمترین مقدار غیر قابل ساده شدن در می‌آیند و البته مخرج به صورت غیر منفی است:

```
julia> 6//9
2//3

julia> -4//8
-1//2

julia> 5//-15
-1//3

julia> -4//-12
1//3
```

این شکل نرمال شده برای یک نسبت صحیح به صورت منحصر به فرد است، بنابراین برابری مقادیر گویا را می‌توان به وسیله چک کردن برابری صورت و مخرج آن ها تست کرد. بوسیله توابع numerator و denominator می‌توان صورت و مخرج استاندارد شده‌ی یک مقدار گویا را از هم جدا کرد:

```
julia> numerator(2//3)
2

julia> denominator(2//3)
3
```

زمانی که عملگرهای استاندارد محاسبه و مقایسه برای مقادیر گویا نیز تعریف شده‌اند، به مقایسه مستقیم صورت و مخرج واقعاً نیازی نیست:

```
julia> 2//3 == 6//9
```

```
true

julia> 2//3 == 9//27
false

julia> 3//7 < 1//2
true

julia> 3//4 > 2//3
true

julia> 2//4 + 1//6
2//3

julia> 5//12 - 1//4
1//6

julia> 5//8 * 3//12
5//32

julia> 6//5 / 10//7
21//25
```

اعداد گویا به شیوه بسیار راحتی می‌توانند به مقادیر اعشاری تبدیل شوند:

```
julia> float(3//4)
0.75
```

تبدیل از گویا به اعشاری، بر طبق هویتِ(تایپ‌هایی که صورت و مخرج دارند) مقادیرِ بدون کسرِ a و b انجام می‌شود. البته به غیر از موردی که a == 0 و b == 0:

```
julia> a = 1; b = 2;

julia> isequal(float(a//b), a/b)
true
```

ساختار مقدار گویای بی‌نهایت نیز قابل قبول است:

```
julia> 5//0
1//0

julia> -3//0
-1//0

julia> typeof(ans)
Rational{Int64}
```

هر چقدر هم سعی کنید که یک مقدار گویای NaN بسازید، با این حال، این مقدار بی‌اعتبار است:

```
julia> 0//0
ERROR: ArgumentError: invalid rational: zero(Int64)//zero(Int64)
Stacktrace:
[...]
```

به طور معمول، بوسیله سیستم ارتقاء زبان برنامه نویسی جولیا، تعاملات با دیگر تایپ‌های عددی آسان شده است:

```
julia> 3//5 + 1
8//5

julia> 3//5 - 0.5
0.09999999999999998
```

```
julia> 2//7 * (1 + 2im)
2//7 + 4//7*im

julia> 2//7 * (1.5 + 2im)
0.42857142857142855 + 0.5714285714285714im

julia> 3//2 / (1 + 2im)
3//10 - 3//5*im

julia> 1//2 + 2im
1//2 + 2//1*im

julia> 1 + 2//3im
1//1 - 2//3*im

julia> 0.5 == 1//2
true

julia> 0.33 == 1//3
false

julia> 0.33 < 1//3
true

julia> 1//3 - 0.33
0.0033333333333332993
```

فصل سوم

۳-۱ برنامه نویسی موازی[1]

رایانش موازی یا محاسبات موازی به اجرای هم‌زمان یک برنامه (که به بخش‌های کوچک‌تری تقسیم شده‌است) بر روی چند پردازنده به منظور دستیابی به سرعت بیشتر گفته می‌شود. ایده اصلی این است که فرایند حل یک مسئله را معمولاً می‌توان به زیروظایف خردتری تقسیم کرد که با اجرای هم‌زمان این زیروظایف و هماهنگ کردن آن ها مسئله اصلی در زمان کوتاه تری حل می‌شود.

برنامه‌نویسی موازی برای استفاده هر چه بهتر از منابع سیستم و افزایش سرعت و کارایی برنامه روی پردازنده‌ها به وجود آمد. در این نوع برنامه‌نویسی، قسمت‌هایی از برنامه اصلی که قابلیت اجرای همزمان[2] را دارند به چند زیربرنامه تقسیم شده و به صورت همزمان روی چند پردازنده یا چند نخ[3] اجرا می‌شوند. قسمتی از برنامه هم که قابلیت اجرای موازی[4] را ندارد به صورت سریال روی یک پردازنده اجرا می‌شود. در واقع تفاوت اصلی برنامه‌سازی ترتیبی و موازی همین امر می‌باشد، اما در پی آن مفاهیم متعددی مطرح می‌شود که اغلب در برنامه‌نویسی معمولی مطرح نبوده و یا ماهیت آن ها با مفهوم متناظر آن در برنامه‌نویسی معمولی کاملاً متفاوت است. یکی از دلایل اصلی استفاده از برنامه‌نویسی موازی، بحث افزایش سرعت اجرای برنامه می‌باشد، که این امر در پردازنده‌های تک هسته‌ای دارای محدودیت‌های زیر می‌باشد:

افزایش تعداد ترانزیستورهای پردازنده برای بدست آوردن سرعت بیشتر، باعث افزایش توان مصرفی خواهد شد که اگر تعداد این ترانزیستورها زیاد باشد، گرمای ایجاد شده ناشی از توان مصرفی پردازنده باعث ذوب شدن آن خواهد شد.

---

[1] Parallel Programming
[2] Concurrent
[3] en:Thread
[4] Parallel

باید به این نکته توجه داشت که اگر پردازنده‌ها بسیار قوی باشند، به دلیل اینکه هر دسترسی به حافظه به چندین کلاک پردازنده نیاز دارد، محدودیت سرعت دسترسی به حافظه باعث خواهد شد پردازنده زمانی را برای نوشتن در یا خواندن از حافظه منتظر بماند و در نتیجه از تمام توان مصرفی آن استفاده نشود.

۳-۲ اصول برنامه نویسی موازی

۱-یافتن موازات کافی در برنامه(قانون آمدال)

طبق قانون آمدال در برنامه‌نویسی موازی، باید طوری برنامه را به دو قسمت موازی و سریال تقسیم کرد، که مقدار سرباری[1] که به سیستم به دلیل تقسیم وظایف میان نخ‌ها/پردازنده‌ها تحمیل می‌شود از مقدار سودی که بدلیل موازی کردن برنامه بدست می‌آوریم کمتر باشد.

۲-دانه دانه کردن

هنگام تقسیم وظایف، باید به اندازه قسمتی از برنامه که قرار است به صورت موازی اجرا شود توجه داشت. به این صورت که در صورت زیاد بودن تعداد آنها و کوچک بودن این قسمت‌ها سربار بسیار زیادی به سیستم تحمیل خواهد شد، و در صورتی که اندازه این قسمت‌ها بسیار بزرگ باشد، در آن صورت این قسمت‌ها تقریباً به صورت سریال اجرا می‌شود که افزایش سرعتی در این حالت نخواهیم داشت.

۳-محلی سازی

سرعت دسترسی به حافظه‌های دارای حجم بالا کم بوده، و سرعت دسترسی به حافظه‌های دارای حجم پایین زیاد می‌باشد. با توجه به این امر، برنامه‌نویس باید طوری عمل کند که الگوریتم‌های موجود در برنامه باید بیشتر کار خود را روی داده‌های موجود در حافظه محلی[2] انجام دهند.

[1] Overhead
[2] Local Memory

۳-عدم تعادل بار[1]

عدم تعادل بار به حالتی گفته می‌شود که در آن بعضی از پردازنده‌ها در برخی زمان‌ها به دلایل زیر کاری را انجام نمی‌دهند:

- موزات ناکافی
- وظایف غیرهم‌اندازه

تعادل بار به دو صورت ایستا[2]، یا پویا[3] در زمان اجرا انجام می‌پذیرد.

۳-۳ امکانات برنامه نویسی موازی

زبان های برنامه نویسی همزمان، کتابخانه ها، رابط های برنامه کاربردی، و مدل های برنامه نویسی موازی (مانند اسکلت الگوریتمی) برای برنامه نویسی موازی کامپیوترها ایجاد شده است. این به طور کلی به چند کلاس بر مبنای حافظه معماری مشترک، حافظه توزیع شده و حافظه توزیع شده مشترک تقسیم می‌شود. زبان های برنامه نویسی حافظه مشترک با دستکاری متغیرهای حافظه مشترک ارتباط برقرار کنید. حافظه توزیع شده از عبور پیام استفاده می‌کنند. ترد های POSIX و OpenMP هردو بیشتر از رابط های برنامه های کاربردی حافظه مشترک استفاده می‌کنند،درحالی که رابط عبور پیام (MPI) بیشتر از رابط های برنامه کاربردی سیستم عبور پیام استفاده می‌کند.

زبان برنامه نویسی جولیا می تواند کارها را به طور کارآمد روی پردازنده گرافیکی انجام دهد. در سال های اخیر GPU در ماشین لرنینگ و دیگر زمینه ها به خاطر قدرت پردازش مفید و قابلیت آن در برنامه نویسی

[1] Load Imbalance
[2] Static
[3] Dynamic

موازی نسبت به قدرت CPU مورد توجه قرار گرفته است. سرعت و مقیاس پذیری زبان برنامه نویسی جولیا آن را به اولین انتخاب برای ترکیب با GPU می کند. در حال حاظر، جولیا می تواند برای اجرای توابع در GPU مانند CPU از CUDAnative.jl استفاده کند.

۳-۴ نمونه برنامه های نوشته شده با برنامه نویسی جولیا

۱-برنامه دریافت عدد از کاربر و گرفتن جذر آن

```
s = readline()
s = parse(Float64,s)
println(sqrt(s))
```

۲- برنامه محاسبه حجم کره

```
function sphere_vol(r)
   return 4/3*pi*r^3
end
vol = sphere_vol(۳)
println(vol)
```

منابع

اخلاقی نیا، ع. ر. ۱۳۹۹. تفکر برنامه نویسی Julia، انتشارات ناقوس، تهران، ایران. ۳۳۲ص.

Huo, Z.; Mei, G.; Casolla, G. & Giampaolo, F. 2020. Designing an efficient parallel spectral clustering algorithm on multi-core processors in Julia. Journal of Parallel and Distributed Computing 138: 211–221.

Xiong, H.; Yin, Z. Y. & Nicot, F. 2020. Programming a micro-mechanical model of granular materials in Julia. Advances in Engineering Software 145:102816.

Voulgaris, Z. 2016. Julia for Data Science. Technics Publictions, United State America. 346 p.

Gao, K.; Mei, G.; Piccialli, F.; Cuomo, S.; Tu, J. & Huo, Z. 2020. Julia language in machine learning: Algorithms, applications, and open issues. Computer Science Review 37: 100254.

Vaiud, G.; Chen, Y.; Bayol, B. & Cournede, P. H. 2016. A Comparison of a Generic MCMC-Based Algorithm for Bayesian Estimation in C++, R and Julia. Application to Plant Growth Modeling. CentraleSupélec, Laboratory MAS, 92290 Châtenay–Malabry, France. 1-13.

Blaom, A. D.; Kiraly, F.; Lienart, T.; Simillides, Y.; Arenas, D. & Vollmer, S. J. 2020. MLJ: A Julia package for composable machine learning. Journal of Open Source Software 5(55): 1-7.

Innes, M. 2018. Flux: Elegant machine learning with Julia. Journal of Open Source Software 3(25): 602.

Tim, B.; Foket, C. & DeSutter. B. 2017. Effective Extensible Programming: Unleashing Julia on GPUs. arXiv abs/11712.03112.