ارائه‌ی یک قالب بهینه برای کدنویسی با فرترن

مصطفی محمدرضایی[۱*]، امیرحسین سعیدی‌نیا[۲]، علی پارسافر[۳]، علیرضا معتضدیان[۴]، ایمان خزرک[۵]، هادی عسکری[۶]

۱-دانشگاه شهید چمران، mostafa.mohammadrezaee@gmail.com
۲-دانشگاه صنعت نفت اهواز، a.saeedinia@gmail.com
۳- پژوهش‌سرای فارابی، ali_parsafar1357@yahoo.com
۴- پژوهش‌سرای فارابی، alireza.motazedian@gmail.com
۵- پژوهش‌سرای فارابی، iman.khazrak.ik@gmail.com
۶- پژوهش‌سرای فارابی، hadi.askari1981@gmail.com

چکیده فارسی

نوشتن یک کد کارآمد برای حل عددی یک مسئله، نکات و پیچیدگی‌های زیادی دارد. از این‌رو سال‌ها تجربه‌ی کدنویسی لازم است تا بتوان خطاهای نحوی برنامه را اصلاح کرد و نتایج عددی درست را استخراج نمود. ابتکار ما در این مقاله، ارائه‌ی قالبی است که یک کدنویس را قادر می‌سازد تا از ایجاد بسیاری از خطاهای پربسامد جلوگیری کند و به آسانی و با دقت بیشتر جواب‌های مطلوب را به دست آورد.

واژه‌های کلیدی: کدنویسی، فرترن، قالب

مقدمه

در عنوان این مقاله بر کدنویسی با فرترن تاکید شده‌است. اما استفاده از این قالب در زبان‌های برنامه‌نویسی دیگر نیز راهگشاست. زیرا یک کدنویس با تجربه می‌داند که بخش عمده‌ی یک کد، طراحی الگوریتم آن بوده و این خود، مستقل از زبان به کار رفته برای نوشتن کد است. این قالب در دوازده گام به صورت خطوط توضیحی ـ با علامت ! در اول خط ـ برنامه را به قسمت‌های جدا از هم تقسیم می‌کند. در این مقاله سعی شده‌است به طور مختصر، مزیت‌های هر گام توضیح داده‌شود. بهتر است این خطوط توضیحی، مانند تابلوهایی، در برنامه کپی شوند و در نهایت طبق تابلوهای راهنما، کد مورد نظر گام به گام نوشته شود. این روش توسط تعدادی از دانشجویان ـ که در سطح‌های مختلفی از برنامه‌نویسی بوده‌اند ـ آزموده شده‌است. درنهایت کارآمدی این قالب ما را بر آن داشت تا در جهت ارتقای سرعت و دقت کدنویسی دیگر دانش‌پژوهان، نسبت به معرفی و انتشار آن اقدام کنیم. علاقمندان می‌توانند برنامه‌های نوشته‌شده با این قالب را از مؤلف دریافت کنند.

معرفی گام‌های برنامه

گام اول) ثبت مشخصات برنامه و برنامه‌نویسان:

شایسته است کدنویسان در ابتدای هر برنامه خود را معرفی کرده و راه ارتباطی‌شان را ثبت نمایند. این کار، گاه باعث می‌شود ارتباطات علمی ارزشمندی با سایر محققان ایجاد شود و به نوعی مؤلفان خود را با این روش معرفی کرده و توانایی خود را در جهت جلب همکاری سایرین نشان‌دهند. همچنین نوشتن توضیحاتی در مورد برنامه، تاریخ نوشتن و بازبینی آن برای مراجعات بعدی ضروری است[۱].

```
!*****************************************************************************************
!     Mostafa Mohammad-Rezaee (MM), Independent researchers
!     Mostafa.mohammadrezaee@gmail.com
!
!     "General Structure.F90" is a suggested structure to numerical computer programming.
!
!                Originally Written: 30-Des-2013   by MM
!                      Last revised: 22-Jan-2014   by MM
!*****************************************************************************************
```

گام دوم) تعریف متغیرها:

در ظاهر، کدنویسی با Matlab که در آن نیازی به تعریف متغیرها نیست، راحت‌تر به نظر می‌رسد. اما تعریف متغیرها در زبان‌های سطح پایین و استفاده از فرمان implicit none ـ که پیش فرض‌های فرترن را از کار می‌اندازد ـ یکی از امتیازهای اصلی برنامه به شمار می‌رود[۱]. تعریف بهینه‌ی تعداد درایه‌های یک آرایه در جهت پیشگیری از اشکال insufficient virtual memory و همچنین افزایش سرعت اجرای برنامه یکی از این برتری‌هاست.

```
!*****************************************************************************************
!                              Variables Definition
!*****************************************************************************************
```

گام سوم) اختصاص‌دادن مقدار اولیه‌ی صفر به متغیرها:

متغیرهای مقداردهی نشده می‌توانند یک مشکل جدی ایجاد کنند. بنابراین تمام متغیرها پیش از استفاده، باید مقداردهی شوند، زیرا در غیر این‌صورت کامپایلر، داده‌های باقی‌مانده از برنامه‌ی قبلی که همان بخش از حافظه را اشغال کرده‌است، گزارش می‌کند. [۲].

```
!*****************************************************************************************
!                              Giving Zero to variables
!*****************************************************************************************
```

گام چهارم) گرفتن مقایر ورودی از کاربر:

متغیرهایی که در اجراهای متوالی برنامه، قراراست مقادیر مختلفی داشته‌باشند و هدف برنامه بررسی تغییر آن پارامترهاست، بهتر است توسط کاربر مقداردهی شوند.

```
!*****************************************************************************************
!                                      Inputs
!*****************************************************************************************
```

یک تکنیک اینست که هر متغیر به دو صورت عددی و کاراکتری دریافت شود تا از فرمت عددی آن برای محاسبات و از فرمت کاراکتری آن برای ساختن نام فایل استفاده کنیم.در مثال زیر متغیرهای E و freq عدد حقیقی ولی متغیرهای EE و freqf همان مقادیر ولی به صورت کاراکتری توسط کامپایلر دریافت می‌شوند.

```
Real   E , freq
Character*30   EE, freqf, filename1

write(*,'(/,2x,a,\)') '          Enter the Energy value  : '
 read(*,*)E
write(*,'(/,2x,a,\)') '                          Again  : '
 read(*,*)EE

write(*,'(/,2x,a,\)') '        Enter the frequency value  : '
 read(*,*)freq
write(*,'(/,2x,a,\)') '                          Again  : '
 read(*,*)freqf
```

گام پنجم) اختصاص‌دادن نام فایل‌ها با استفاده از اطلاعات ورودی یا مشخصات مسأله:

این بخش نکته‌ی باریکی دارد که اگر به درستی از آن استفاده کنیم، اولاً سرعت کار با فایل‌های خروجی بالا می‌رود، دوم اینکه اطلاعات خروجی به خوبی دسته‌بندی می‌شوند. برای این‌کار لازم است نام فایل را از اطلاعات ورودی بسازیم.

```
!*****************************************************************************************
!                    Determination of Filenames and Opening files
!*****************************************************************************************
```

در ضمن اگر خروجی‌های ما به صورت نمودار باشند بهتر است فایل با پسوند plt ـ مربوط به نرم‌افزار tecplot ـ ذخیره شود. این نرم‌افزار علاوه بر قابلیت‌های متعدد این امتیاز را دارد که فقط با کلیک کردن روی آیکن فایل، نمودار رسم می‌شود.عملگر// دو عبارت کاراکتری را به هم می‌چسباند و فرمان trim( ) کاراکترهای اشغال نشده‌ی یک میدان کاراکتری را حذف می‌کند[۳]. (مثال: نمودار دما بر حسب زمان برای انرژی و فرکانس مشخص پالس)

```
filenameTt = 'E'//trim(EE)//' f'//trim(freqf)//' Tt.plt'
open(1,file=filenameTt)
```

گام ششم) واردکردن مقادیر ثابت به برنامه:

بهتر است تمام مقادیر برنامه در این قسمت متمرکز شوند. همچنین باید چیدمان آنها نظمی خوش‌خوان و معنادار داشته باشد تا کنترل آنها به آسانی انجام شود.

```
!*****************************************************************************************
!                                Constants
!*****************************************************************************************
```

گام هفتم) تعیین‌کردن تعداد درایه‌های هر بعدِ آرایه به صورت بهینه با استفاده از مقادیر ورودی و مقادیر ثابت:

هر چه تعداد درایه‌های یک آرایه کمتر باشد احتمال کاهش زمان اجرای برنامه و برخورد با خطاهای memory کمتر است. بنابرین لازم است آرایه‌ها به صورت بهینه allocate شوند. در کل بهتر است برای حل یک مسئله از آرایه استفاده نکنیم مگر آنکه واقعاً مورد نیاز باشد[۲].

```
!*****************************************************************************************
!                              Arrays Allocation
!*****************************************************************************************
```

گام هشتم) اختصاص‌دادن مقدار اولیه‌ی صفر به تمام درایه‌های آرایه‌ها:

آرایه‌ها نیز باید مانند متغیرها، مقداردهی اولیه شوند. برای این‌کار بهتر است مقدار صفر توسط فرمان forall به همه‌ی درایه‌ها اختصاص داده‌شود[۲].

```
!*****************************************************************************
!                    Giving Zero to Arrays
!*****************************************************************************
```

گام نهم) چاپ مقادیر ورودی و مقادیر ثابت برنامه:

لازم است کاربر پیش از اجرای بخش اصلی برنامه، نسبت به درست بودن تمام مقادیری که با آنها برنامه را تغذیه می‌کند، مطمئن شود. در این راستا بهتر است نام متغیرها و مقدار آنها به طور شایسته‌ای روبروی هم چاپ شوند تا کاربر به راحتی بتواند آنها را کنترل کند.

```
!*****************************************************************************
!                    Printing Constants and Inputs
!*****************************************************************************
```

گام دهم) بخش اصلی برنامه:

در این بخش شرایط مرزی و الگوریتم حل مسأله نوشته می‌شود.

```
!*****************************************************************************
!                    Main Block of the Program
!*****************************************************************************
```

گام یازدهم) چاپ نتایج:

هر چند چاپ نتایج به عنوان یک بخش مجزا معرفی شده‌است، اما گاهی لازم است اطلاعات در همان بخش اصلی برنامه مستقیماً در فایل‌ها ذخیره شوند. اگر در برنامه محل دریافت خروجی‌ها هوشمندانه انتخاب شود، می‌تواند در اغلب موارد از بروز پیغام insufficient virtual memory جلوگیری کند و سرعت اجرای برنامه را گاهی ده‌ها مرتبه بالا ببرد.

```
!*****************************************************************************
!                    Printing Results
!*****************************************************************************
```

گام دوازدهم) بستن فایل‌های بازشده و خاتمه‌دادن به برنامه:

در نگاه اول به نظر می‌رسد، بازنگهداشتن فایل‌هایی که در ابتدای برنامه باز شده مشکلی ایجاد نمی‌کند ولی تجربه نشان داده، برای پیشگیری از اشکالات ناشی از تداخل برنامه در اجراهای مکرر، بهتر است فایل‌ها در انتهای کار بسته شوند [۳].

```
!*****************************************************************************
!                    Closing Files and Ending the Program
!*****************************************************************************
```

نتیجه‌گیری

در این مقاله نوسیندگان با استفاده از تجربه‌ی چندین ساله‌ی خود سعی بر آن داشته‌اند قالبی برای نوشتن یک برنامه ارائه کنند که پیروی از آن، کاربران -از حرفه‌ای تا مبتدی- را قادر خواهد ساخت که کدهای بهینه‌ای بنویسند و از بسیاری خطاهای پربسامد پرهیز کنند. البته برنامه‌نویسی مدولار هم به نوعی بخشی از این اهداف را دنبال می‌کند اما ما بر این باوریم که این قالب بسیار راحت‌تر و کارآمدتر است.

مراجع

[۱] یان‌م، اسمیت؛ «*برنامه‌نویسی در فرترن ۹۰*» جهاددانشگاهی واحد تهران؛ صفحه ۷ .

[۲] جی چپمن، استفان؛ «*کتاب آموزشی Fortran 90/95*» ناقوس؛ صفحه‌های ۶۸ و ۳۰۵ .

[۳] صفاری، حمید؛ «*کتاب آموزشی Visual Fortran*» مؤلف؛ صفحه‌های ۲۵ و ۴۶ .