اصول برنامه‌نویسی کامپیوتر

فصل سوم: زیرالگوریتم و زیرفلوچارت

پیش به سوی حل مسائل پیچیده‌تر

محمّد فرشی

بخش علوم کامپیوتر - دانشکدهٔ علوم ریاضی - دانشگاه یزد

۱۴۰۳-۱

حل مسائل پیچیده‌تر

اگر مسائل مورد نظر، بسیار پیچیده باشند چطور؟

- فرض کنید بخواهیم نرم‌افزاری جهت پردازش تصاویر و یا تولید برنامه‌های پویانمایی طراحی کنیم.
- آن را به‌عنوان یک مسئله واحد درنظر گرفته و برای حل آن تلاش کنیم با روند سخت و نفس‌گیری مواجه خواهیم شد.
- یک راه‌حل برای این مشکل، تشکیل تیمی از افراد است برای حل مسئله.
- لازمهٔ کار تیمی آن است که مسئله، به قسمت‌هایی حتی‌الامکان مستقل تجزیه شود و هرقسمت توسط یکی یا چند نفر از اعضای تیم حل شود و درنهایت، پس از تکمیل قسمت‌ها و بررسی درستی عملکرد آن‌ها، کار به‌هم وصل کردن نتایج و ساختن محصول نهایی صورت گیرد.
- که یک مسئله را به شکل‌های مختلف می‌توان تجزیه کرد و روند تجزیه عملاً فی نفسه مهم است.
- مزیت تقسیم مسائل در خطایابی.

حل مسائل پیچیده‌تر

اگر مسائل مورد نظر، بسیار پیچیده باشند چطور؟

- فرض کنید بخواهیم نرم‌افزاری جهت پردازش تصاویر و یا تولید برنامه‌های پویانمایی طراحی کنیم.
- آن را به‌عنوان یک مسئله واحد درنظر گرفته و برای حل آن تلاش کنیم با روند سخت و نفس‌گیری مواجه خواهیم شد.
- یک راه‌حل برای این مشکل، تشکیل تیمی از افراد است برای حل مسئله.
- لازمهٔ کار تیمی آن است که مسئله، به قسمت‌هایی حتی‌الامکان مستقل تجزیه شود و هرقسمت توسط یکی یا چند نفر از اعضای تیم حل شود و درنهایت، پس از تکمیل قسمت‌ها و بررسی درستی عملکرد آن‌ها، کار به‌هم وصل کردن نتایج و ساختن محصول نهایی صورت گیرد.
- که یک مسئله را به شکل‌های مختلف می‌توان تجزیه کرد و روند تجزیه عملاً فی نفسه مهم است.
- مزیت تقسیم مسائل در خطایابی.

حل مسائل پیچیده‌تر

اگر مسائل مورد نظر، بسیار پیچیده باشند چطور؟

- فرض کنید بخواهیم نرم‌افزاری جهت پردازش تصاویر و یا تولید برنامه‌های پویانمایی طراحی کنیم.
- آن را به‌عنوان یک مسئله واحد درنظر گرفته و برای حل آن تلاش کنیم با روند سخت و نفس‌گیری مواجه خواهیم شد.
- یک راه‌حل برای این مشکل، تشکیل تیمی از افراد است برای حل مسئله.
- لازمهٔ کار تیمی آن است که مسئله، به قسمت‌هایی حتی‌الامکان مستقل تجزیه شود و هرقسمت توسط یکی یا چند نفر از اعضای تیم حل شود و درنهایت، پس از تکمیل قسمت‌ها و بررسی درستی عملکرد آن‌ها، کار به‌هم وصل کردن نتایج و ساختن محصول نهایی صورت گیرد.
- که یک مسئله را به شکل‌های مختلف می‌توان تجزیه کرد و روند تجزیه عملاً فی نفسه مهم است.
- مزیت تقسیم مسائل در خطایابی.

حل مسائل پیچیده‌تر

اگر مسائل مورد نظر، بسیار پیچیده باشند چطور؟

- فرض کنید بخواهیم نرم‌افزاری جهت پردازش تصاویر و یا تولید برنامه‌های پویانمایی طراحی کنیم.
- آن را به‌عنوان یک مسئله واحد درنظر گرفته و برای حل آن تلاش کنیم با روند سخت و نفس‌گیری مواجه خواهیم شد.
- یک راه‌حل برای این مشکل، تشکیل تیمی از افراد است برای حل مسئله.
- لازمهٔ کار تیمی آن است که مسئله، به قسمت‌هایی حتی‌الامکان مستقل تجزیه شود و هرقسمت توسط یکی یا چند نفر از اعضای تیم حل شود و درنهایت، پس از تکمیل قسمت‌ها و بررسی درستی عملکرد آن‌ها، کار به‌هم وصل کردن نتایج و ساختن محصول نهایی صورت گیرد.
- که یک مسئله را به شکل‌های مختلف می‌توان تجزیه کرد و روند تجزیه عملاً فی نفسه مهم است.
- مزیت تقسیم مسائل در خطایابی.

حل مسائل پیچیده‌تر

اگر مسائل مورد نظر، بسیار پیچیده باشند چطور؟

- فرض کنید بخواهیم نرم‌افزاری جهت پردازش تصاویر و یا تولید برنامه‌های پویانمایی طراحی کنیم.
- آن را به‌عنوان یک مسئله واحد درنظر گرفته و برای حل آن تلاش کنیم با روند سخت و نفس‌گیری مواجه خواهیم شد.
- یک راه‌حل برای این مشکل، تشکیل تیمی از افراد است برای حل مسئله.
- لازمهٔ کار تیمی آن است که مسئله، به قسمت‌هایی حتی‌الامکان مستقل تجزیه شود و هرقسمت توسط یکی یا چند نفر از اعضای تیم حل شود و درنهایت، پس از تکمیل قسمت‌ها و بررسی درستی عملکرد آن‌ها، کار به‌هم وصل کردن نتایج و ساختن محصول نهایی صورت گیرد.
- که یک مسئله را به شکل‌های مختلف می‌توان تجزیه کرد و روند تجزیه عملاً فی نفسه مهم است.
- مزیت تقسیم مسائل در خطایابی.

حل مسائل پیچیده‌تر

اگر مسائل مورد نظر، بسیار پیچیده باشند چطور؟

- فرض کنید بخواهیم نرم‌افزاری جهت پردازش تصاویر و یا تولید برنامه‌های پویانمایی طراحی کنیم.
- آن را به‌عنوان یک مسئله واحد درنظر گرفته و برای حل آن تلاش کنیم با روند سخت و نفس‌گیری مواجه خواهیم شد.
- یک راه‌حل برای این مشکل، تشکیل تیمی از افراد است برای حل مسئله.
- لازمهٔ کار تیمی آن است که مسئله، به قسمت‌هایی حتی‌الامکان مستقل تجزیه شود و هرقسمت توسط یکی یا چند نفر از اعضای تیم حل شود و درنهایت، پس از تکمیل قسمت‌ها و بررسی درستی عملکرد آن‌ها، کار به‌هم وصل کردن نتایج و ساختن محصول نهایی صورت گیرد.
- که یک مسئله را به شکل‌های مختلف می‌توان تجزیه کرد و روند تجزیه عملاً فی نفسه مهم است.
- مزیت تقسیم مسائل در خطایابی.

حل مسائل پیچیده‌تر

مثال:

- مسئله: مطلوب است تعداد اعداد اول بین ۲ تا ۱۰۰۰.
- روش: بررسی تمام اعداد بین ۲ تا ۱۰۰۰.
- زیرمسئله: تشخیص اول بودن عدد.
- این زیرمسئله یک عدد را به عنوان پارامتر، و نه ورودی و از طریق کاربر، دریافت می‌کند و بررسی می‌کند اول است یا خیر. نتیجهٔ آزمایش به عنوان نتیجه برگردانده می‌شود.
- دقت شود که اولاً، ورودی را از صفحه کلید یا کاربر دریافت نمی‌کند و ثانیاً نتیجهٔ آزمایش در خروجی چاپ نمی‌شود بلکه به فردی (در اینجا الگوریتم) که این زیرمسئله را استفاده کرده است اعلام می‌شود.
- استفاده از یک زیرالگوریتم را اصطلاحاً فراخوانی زیرالگوریتم و برگرداندن نتیجه را بازگشت و اعلام نتیجه را برگرداندن نتیجه گویند.

حل مسائل پیچیده‌تر

مثال:

- مسئله: مطلوب است تعداد اعداد اول بین ۲ تا ۱۰۰۰.
- روش: بررسی تمام اعداد بین ۲ تا ۱۰۰۰.
- زیرمسئله: تشخیص اول بودن عدد.
- این زیرمسئله یک عدد را به عنوان پارامتر، و نه ورودی و از طریق کاربر، دریافت می‌کند و بررسی می‌کند اول است یا خیر. نتیجهٔ آزمایش به عنوان نتیجه برگردانده می‌شود.
- دقت شود که اولاً، ورودی را از صفحه کلید یا کاربر دریافت نمی‌کند و ثانیاً نتیجهٔ آزمایش در خروجی چاپ نمی‌شود بلکه به فردی (در اینجا الگوریتم) که این زیرمسئله را استفاده کرده است اعلام می‌شود.
- استفاده از یک زیرالگوریتم را اصطلاحاً فراخوانی زیرالگوریتم و برگرداندن نتیجه را بازگشت و اعلام نتیجه را برگرداندن نتیجه گویند.

حل مسائل پیچیده‌تر

مثال:

- مسئله: مطلوب است تعداد اعداد اول بین ۲ تا ۱۰۰۰.
- روش: بررسی تمام اعداد بین ۲ تا ۱۰۰۰.
- زیرمسئله: تشخیص اول بودن عدد.
- این زیرمسئله یک عدد را به عنوان پارامتر، و نه ورودی و از طریق کاربر، دریافت می‌کند و بررسی می‌کند اول است یا خیر. نتیجهٔ آزمایش به عنوان نتیجه برگردانده می‌شود.
- دقت شود که اولاً، ورودی را از صفحه کلید یا کاربر دریافت نمی‌کند و ثانیاً نتیجهٔ آزمایش در خروجی چاپ نمی‌شود بلکه به فردی (در اینجا الگوریتم) که این زیرمسئله را استفاده کرده است اعلام می‌شود.
- استفاده از یک زیرالگوریتم را اصطلاحاً فراخوانی زیرالگوریتم و برگرداندن نتیجه را بازگشت و اعلام نتیجه را برگرداندن نتیجه گویند.

حل مسائل پیچیده‌تر

مثال:

- مسئله: مطلوب است تعداد اعداد اول بین ۲ تا ۱۰۰۰.
- روش: بررسی تمام اعداد بین ۲ تا ۱۰۰۰.
- زیرمسئله: تشخیص اول بودن عدد.
- این زیرمسئله یک عدد را به عنوان پارامتر، و نه ورودی و از طریق کاربر، دریافت می‌کند و بررسی می‌کند اول است یا خیر. نتیجهٔ آزمایش به عنوان نتیجه برگردانده می‌شود.
- دقت شود که اولاً، ورودی را از صفحه کلید یا کاربر دریافت نمی‌کند و ثانیاً نتیجهٔ آزمایش در خروجی چاپ نمی‌شود بلکه به فردی (در اینجا الگوریتم) که این زیرمسئله را استفاده کرده است اعلام می‌شود.
- استفاده از یک زیرالگوریتم را اصطلاحاً فراخوانی زیرالگوریتم و برگرداندن نتیجه را بازگشت و اعلام نتیجه را برگرداندن نتیجه گویند.

حل مسائل پیچیده‌تر

مثال:

- مسئله: مطلوب است تعداد اعداد اول بین ۲ تا ۱۰۰۰.
- روش: بررسی تمام اعداد بین ۲ تا ۱۰۰۰.
- زیرمسئله: تشخیص اول بودن عدد.
- این زیرمسئله یک عدد را به عنوان پارامتر، و نه ورودی و از طریق کاربر، دریافت می‌کند و بررسی می‌کند اول است یا خیر. نتیجهٔ آزمایش به عنوان نتیجه برگردانده می‌شود.
- دقت شود که اولاً، ورودی را از صفحه کلید یا کاربر دریافت نمی‌کند و ثانیاً نتیجهٔ آزمایش در خروجی چاپ نمی‌شود بلکه به فردی (در اینجا الگوریتم) که این زیرمسئله را استفاده کرده است اعلام می‌شود.
- استفاده از یک زیرالگوریتم را اصطلاحاً فراخوانی زیرالگوریتم و برگرداندن نتیجه را بازگشت و اعلام نتیجه را برگرداندن نتیجه گویند.

حل مسائل پیچیده‌تر

مثال:

- مسئله: مطلوب است تعداد اعداد اول بین ۲ تا ۱۰۰۰.
- روش: بررسی تمام اعداد بین ۲ تا ۱۰۰۰.
- زیرمسئله: تشخیص اول بودن عدد.
- این زیرمسئله یک عدد را به عنوان پارامتر، و نه ورودی و از طریق کاربر، دریافت می‌کند و بررسی می‌کند اول است یا خیر. نتیجهٔ آزمایش به عنوان نتیجه برگردانده می‌شود.
- دقت شود که اولاً، ورودی را از صفحه کلید یا کاربر دریافت نمی‌کند و ثانیاً نتیجهٔ آزمایش در خروجی چاپ نمی‌شود بلکه به فردی (در اینجا الگوریتم) که این زیرمسئله را استفاده کرده است اعلام می‌شود.
- استفاده از یک زیرالگوریتم را اصطلاحاً فراخوانی زیرالگوریتم و برگرداندن نتیجه را بازگشت و اعلام نتیجه را برگرداندن نتیجه گویند.

حل مسائل پیچیده‌تر

زیرالگوریتم:

- برای هر زیرالگوریتم یک نام تخصیص داده می‌شود.
- اگر زیرالگوریتم باید مقدار یا مقادیری را از الگوریتم اصلی بگیرد، آنها را در پرانتزی جلوی نام زیرالگوریتم می‌آوریم و آنها را پارامتر(های) زیرالگوریتم می‌نامیم.
- نتیجه اجرای یک زیرالگوریتم، می‌تواند به صورت یک یا چند مقدار به الگوریتم فراخوانندهٔ زیرالگوریتم برگشت داده شود.
- ممکن است یک زیرالگوریتم هیچ مقدار بازگشتی نداشته باشد.

حل مسائل پیچیده‌تر

زیرالگوریتم:

- برای هر زیرالگوریتم یک نام تخصیص داده می‌شود.
- اگر زیرالگوریتم باید مقدار یا مقادیری را از الگوریتم اصلی بگیرد، آنها را در پرانتزی جلوی نام زیرالگوریتم می‌آوریم و آنها را پارامتر(های) زیرالگوریتم می‌نامیم.
- نتیجه اجرای یک زیرالگوریتم، می‌تواند به صورت یک یا چند مقدار به الگوریتم فراخوانندهٔ زیرالگوریتم برگشت داده شود.
- ممکن است یک زیرالگوریتم هیچ مقدار بازگشتی نداشته باشد.

حل مسائل پیچیده‌تر

زیرالگوریتم:

- برای هر زیرالگوریتم یک نام تخصیص داده می‌شود.
- اگر زیرالگوریتم باید مقدار یا مقادیری را از الگوریتم اصلی بگیرد، آنها را در پرانتزی جلوی نام زیرالگوریتم می‌آوریم و آنها را پارامتر(های) زیرالگوریتم می‌نامیم.
- نتیجه اجرای یک زیرالگوریتم، می‌تواند به صورت یک یا چند مقدار به الگوریتم فراخوانندهٔ زیرالگوریتم برگشت داده شود.
- ممکن است یک زیرالگوریتم هیچ مقدار بازگشتی نداشته باشد.

حل مسائل پیچیده‌تر

زیرالگوریتم:

- برای هر زیرالگوریتم یک نام تخصیص داده می‌شود.
- اگر زیرالگوریتم باید مقدار یا مقادیری را از الگوریتم اصلی بگیرد، آنها را در پرانتزی جلوی نام زیرالگوریتم می‌آوریم و آنها را پارامتر(های) زیرالگوریتم می‌نامیم.
- نتیجه اجرای یک زیرالگوریتم، می‌تواند به صورت یک یا چند مقدار به الگوریتم فراخوانندهٔ زیرالگوریتم برگشت داده شود.
- ممکن است یک زیرالگوریتم هیچ مقدار بازگشتی نداشته باشد.

مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

الگوریتم:

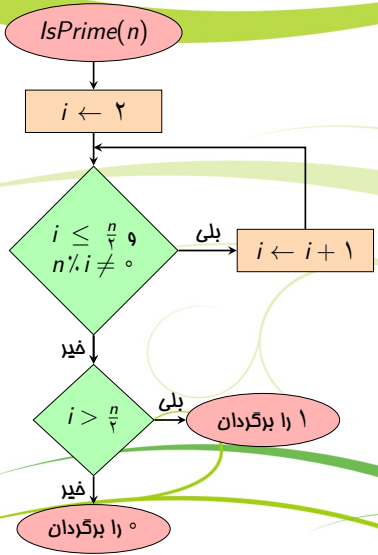
۱ شروع زیرالگوریتم $IsPrime(n)$
۲ $i \leftarrow ۲$
۳ اگر $i \leq \frac{n}{۲}$ و $n \,\%\, i \neq ۰$ آنگاه
۴ $i \leftarrow i + ۱$
۵ به مرحلهٔ ۳ برو
۶ پایان اگر
۷ اگر $i > \frac{n}{۲}$ آنگاه
۸ ۱ را برگردان
۹ درغیراینصورت
۱۰ ۰ را برگردان
۱۱ پایان اگر

مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

الگوریتم:

۱ شروع زیرالگوریتم $IsPrime(n)$
۲ $i \leftarrow ۲$
۳ اگر $i \leq \frac{n}{۲}$ و $n \,\%\, i \neq ۰$ آنگاه
۴ $i \leftarrow i + ۱$
۵ به مرحلهٔ ۳ برو
۶ پایان اگر
۷ اگر $i > \frac{n}{۲}$ آنگاه
۸ ۱ را برگردان
۹ درغیراینصورت
۱۰ ۰ را برگردان
۱۱ پایان اگر



مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

الگوریتم:

۱ شروع زیرالگوریتم $IsPrime(n)$
۲ $i \leftarrow 2$
۳ اگر $i \leq \frac{n}{2}$ و $n \% i \neq 0$ آنگاه
۴ $i \leftarrow i + 1$
۵ به مرحلهٔ ۳ برو
۶ پایان اگر
۷ اگر $i > \frac{n}{2}$ آنگاه
۸ ۱ را برگردان
۹ درغیراینصورت
۱۰ ۰ را برگردان
۱۱ پایان اگر

برنامه:

```
1  int IsPrime(unsigned int n)
2  {
3    if(n == 1)
4      return 0;
5    int i = 2;
6    while(i<=n/2 && n%i !=0)
7         i++;
8    if(i> n/2)
9         return 1;
10     else
11        return 0;
12 }
```

مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

الگوریتم:

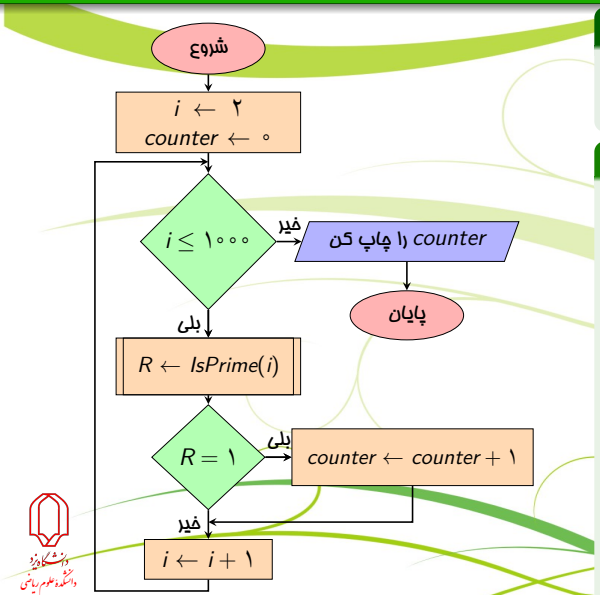
۱ شروع
۲ $i \leftarrow ۲$، $counter \leftarrow ۰$
۳ اگر $i \leq ۱۰۰۰$ آنگاه
۴ $R \leftarrow \text{IsPrime}(i)$
۵ اگر $R = ۱$ آنگاه
۶ $counter \leftarrow counter + ۱$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ $counter$ را چاپ کن
۱۲ پایان

مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ $i \leftarrow ۲$، $counter \leftarrow ۰$
۳ اگر $i \leq ۱۰۰۰$ آنگاه
۴ $R \leftarrow \text{IsPrime}(i)$
۵ اگر $R = ۱$ آنگاه
۶ $counter \leftarrow counter + ۱$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ $counter$ را چاپ کن
۱۲ پایان



مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ $i \leftarrow ۲$، $counter \leftarrow ۰$
۳ اگر $i \leq ۱۰۰۰$ آنگاه
۴ $R \leftarrow \text{IsPrime}(i)$
۵ اگر $R = ۱$ آنگاه
۶ $counter \leftarrow counter + ۱$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ $counter$ را چاپ کن
۱۲ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int IsPrime(unsigned int n)
4  {
5    int i = 2;
6    if(n == 1)
7         return 0;
8    while(i<=n/2 && n%i !=0)
9         i++;
10   if(i> n/2)
11        return 1;
12     else
13        return 0;
14 }
15 int main()
16 {
17   unsigned int i,counter=0,R;
18   for(i = 2; i <= 1000; i++)
19     {
20        R=IsPrime(i);
21        if (R == 1)
22            counter++;
23   }
24   cout << "# of Primes:" <<counter;
25   return 0;
26 }
```

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

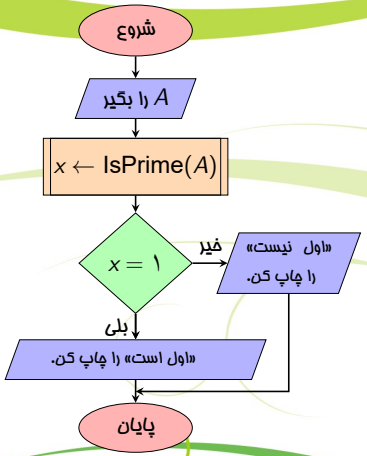
۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان



استفاده مجدد!

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int IsPrime(unsigned int n)
4  {
5      int i = 2;
6    if(n ==1)
7      return 0;
8      while(i<=n/2 && n%i !=0)
9          i++;
10     if(i> n/2)
11         return 1;
12     else
13         return 0;
14 }
15 int main()
16 {
17     unsigned int A,x;
18     cout << "Please enter a positive
           integer:";
19     cin >> A;
20     x = IsPrime(A);
21     if(x == 1)
22         cout<<"Is Prime.";
23     else
24         cout<<"Not Prime.";
25     return 0;
26 }
```

مثال ۴

الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.

مثال ۴

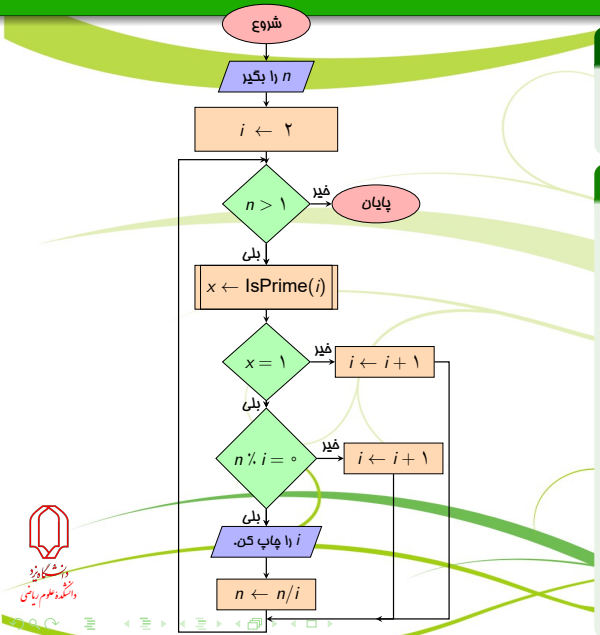
الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow ۲$
۴ اگر $n > ۱$ آنگاه
۵ $x \leftarrow \text{IsPrime}(i)$
۶ اگر $x = ۱$ آنگاه
۷ اگر $n \,\%\, i = ۰$ آنگاه
۸ i را چاپ کن
۹ $n \leftarrow n/i$
۱۰ درغیراینصورت
۱۱ $i \leftarrow i + ۱$
۱۲ پایان اگر
۱۳ درغیراینصورت
۱۴ $i \leftarrow i + ۱$
۱۵ پایان اگر
۱۶ به مرحلهٔ ۴ برو
۱۷ پایان اگر
۱۸ پایان

مثال ۴

الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.



الگوریتم:

```
۱ شروع
۲ n را بگیر
۳ i ← ۲
۴ اگر n > ۱ آنگاه
۵     x ← IsPrime(i)
۶     اگر x = ۱ آنگاه
۷         اگر n ٪ i = ۰ آنگاه
۸             i را چاپ کن
۹             n ← n/i
۱۰        درغیراینصورت
۱۱            i ← i + ۱
۱۲        پایان اگر
۱۳    درغیراینصورت
۱۴        i ← i + ۱
۱۵    پایان اگر
۱۶    به مرحلهٔ ۴ برو
۱۷ پایان اگر
۱۸ پایان
```

مثال ۴

الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow ۲$
۴ اگر $n > ۱$ آنگاه
۵ $x \leftarrow \text{IsPrime}(i)$
۶ اگر $x = ۱$ آنگاه
۷ اگر $n \,\%\, i = ۰$ آنگاه
۸ i را چاپ کن
۹ $n \leftarrow n/i$
۱۰ درغیراینصورت
۱۱ $i \leftarrow i + ۱$
۱۲ پایان اگر
۱۳ درغیراینصورت
۱۴ $i \leftarrow i + ۱$
۱۵ پایان اگر
۱۶ به مرحلهٔ ۴ برو
۱۷ پایان اگر
۱۸ پایان

برنامه:

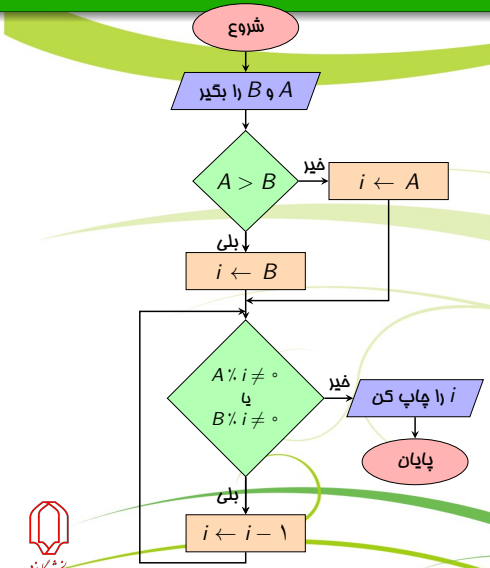
```
#include <iostream>
using namespace std;
int IsPrime(unsigned int n)
{
  int i = 2;
  while(i<=n/2 && n%i !=0)
       i++;
  if(i> n/2)
       return 1;
    else
       return 0;
}
int main()
{
   unsigned int n,i,x;
   cout << "Please enter a positive
        integer:";
   cin >> n;
   for(i=2; n>1; )
   {
      x = IsPrime(i);
      if(x == 1)
         if(n%i == 0)
         {
            cout<<i<<"*";
            n/=i;
         }
         else
            i++;
      else
         i++;
   }
   return 0;
}
```

مثال ۵

فلوچارتی رسم کنید که دو عدد طبیعی را به عنوان ورودی بگیرد و بزرگ‌ترین مقسوم علیه (ب.م.م.) آنها را محاسبه و چاپ کند. (مثال فصل قبل)

الگوریتم:

۱ شروع
۲ A و B را بگیر
۳ اگر $A > B$ آنگاه
۴ | $i \leftarrow B$
۵ درغیراینصورت
۶ | $i \leftarrow A$
۷ پایان اگر
۸ اگر $A \,\%\, i \neq 0$ یا $B \,\%\, i \neq 0$ آنگاه
۹ | $i \leftarrow i - 1$
۱۰ | برو به خط ۸
۱۱ پایان اگر
۱۲ i را چاپ کن
۱۳ پایان



مثال ۶

زیرفلوچارتی رسم کنید که دو عدد طبیعی را به عنوان پارامتر بگیرد و بزرگ‌ترین مقسوم علیه (ب.م.م.) آنها را محاسبه کرده و برگرداند.

مثال ۶

زیرفلوچارتی رسم کنید که دو عدد طبیعی را به عنوان پارامتر بگیرد و بزرگ‌ترین مقسوم علیه (ب.م.م.) آنها را محاسبه کرده و برگرداند.

الگوریتم:

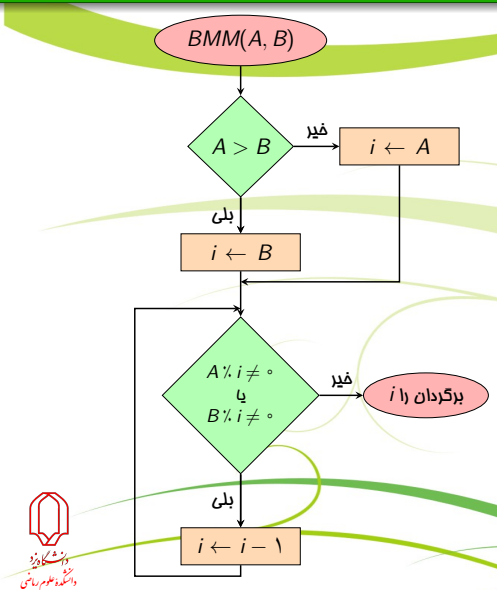
۱ شروع زیرالگوریتم $BMM(A, B)$
۲ اگر $A > B$ آنگاه
۳ $\quad i \leftarrow B$
۴ درغیراینصورت
۵ $\quad i \leftarrow A$
۶ پایان اگر
۷ اگر $i \neq 0$ یا $A \,\%\, i \neq 0$ یا $B \,\%\, i \neq 0$ آنگاه
۸ $\quad i \leftarrow i - 1$
۹ $\quad$ برو به خط ۸
۱۰ پایان اگر
۱۱ i را برگردان

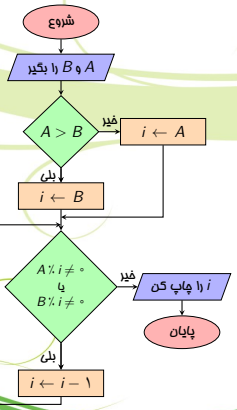
الگوریتم:

۱ شروع
۲ A و B را بگیر
۳ اگر $A > B$ آنگاه
۴ $\quad i \leftarrow B$
۵ درغیراینصورت
۶ $\quad i \leftarrow A$
۷ پایان اگر
۸ اگر $i \neq 0$ یا $A \,\%\, i \neq 0$ یا $B \,\%\, i \neq 0$ آنگاه
۹ $\quad i \leftarrow i - 1$
۱۰ $\quad$ برو به خط ۸
۱۱ پایان اگر
۱۲ i را چاپ کن
۱۳ پایان

مثال ۶

زیرفلوچارتی رسم کنید که دو عدد طبیعی را به عنوان پارامتر بگیرد و بزرگ‌ترین مقسوم علیه (ب.م.م.) آنها را محاسبه کرده و برگرداند.





مثال ۶

زیرفلوچارتی رسم کنید که دو عدد طبیعی را به عنوان پارامتر بگیرد و بزرگ‌ترین مقسوم علیه (ب.م.م.) آنها را محاسبه کرده و برگرداند.

برنامه:

```
#include <iostream>
using namespace std;
int main()
{
    int A,B,i;
    cout<<"Enter two numbers:";
    cin>>A>>B;
    if(A>B)
        i=B;
    else
        i=A;
    while(A%i !=0 || B%i !=0)
        i--;
    cout<<i;
    return 0;
}
```

برنامه:

```
int BMM(int A, int B)
{
    int i;
    if (A > B)
       i=B;
    else
       i=A;
    while ( A % i != 0 || B % i !=0)
       i--;
    return i;
}
```

مثال ۷

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

مثال ۷

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

الگوریتم:

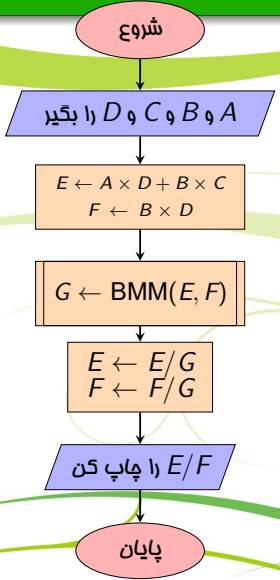
۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow A \times D + B \times C$
۴ $F \leftarrow B \times D$
۵ $G \leftarrow \mathrm{BMM}(E, F)$
۶ $F \leftarrow F/G$ ،$E \leftarrow E/G$
۷ E/F را چاپ کن
۸ پایان

مثال ۷

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow A \times D + B \times C$
۴ $F \leftarrow B \times D$
۵ $G \leftarrow \text{BMM}(E, F)$
۶ $E \leftarrow E/G$، $F \leftarrow F/G$
۷ E/F را چاپ کن
۸ پایان



مثال ۷

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow A \times D + B \times C$
۴ $F \leftarrow B \times D$
۵ $G \leftarrow \mathrm{BMM}(E, F)$
۶ $F \leftarrow F/G$ ،$E \leftarrow E/G$
۷ E/F را چاپ کن
۸ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int BMM(int A, int B)
4  {
5      int i;
6      if (A > B)
7         i=B;
8      else
9         i=A;
10     while ( A % i != 0 || B % i !=0)
11        i--;
12     return i;
13 }
14 int main()
15 {
16     int A,B,C,D,E,F,G;
17     cout << "I want to compute A / B + C /
           D.\n";
18     cin>>A>>B>>C>>D;
19     E=A*D+B*C; F=B*D;
20     G=BMM(E,F);
21     E/=G; F/=G;
22     cout << "The result:" << E << "/" << F;
23     return 0;
24 }
```

مثال ۸

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

مثال ۸

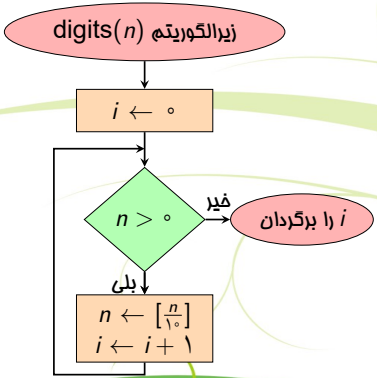
الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع زیرالگوریتم digits(n)
۲ $i \leftarrow ۰$
۳ اگر $n > ۰$ آنگاه
۴ | $n \leftarrow \left[\frac{n}{۱۰}\right]$
۵ | $i \leftarrow i + ۱$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ i را برگردان

مثال ۸

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.



الگوریتم:

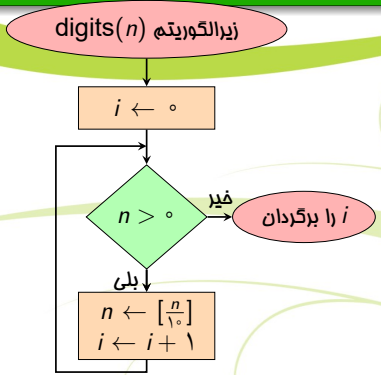
۱ شروع زیرالگوریتم digits(n)
۲ $i \leftarrow 0$
۳ اگر $n > 0$ آنگاه
۴ $\quad n \leftarrow \left[\frac{n}{10}\right]$
۵ $\quad i \leftarrow i + 1$
۶ $\quad$ به مرحلهٔ ۳ برو
۷ پایان اگر
۸ i را برگردان

مثال ۸

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع زیرالگوریتم digits(n)
۲ $i \leftarrow 0$
۳ اگر $n > 0$ آنگاه
۴ $\quad n \leftarrow \left[\frac{n}{10}\right]$
۵ $\quad i \leftarrow i + 1$
۶ $\quad$ به مرحلهٔ ۳ برو
۷ پایان اگر
۸ i را برگردان



برنامه:

```
1 int digits(long int n)
2 {
3   int i;
4   for(i=0; n>0; i++)
5     n/=10;
6   return i;
7 }
```

مثال ۹

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

مثال ۹

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ **شروع** زیرالگوریتم $concat2(x, y)$

۲ $i \leftarrow \text{digits}(x)$

۳ $A \leftarrow y \times ۱۰^{i} + x$

۴ A را برگردان

مثال ۹

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

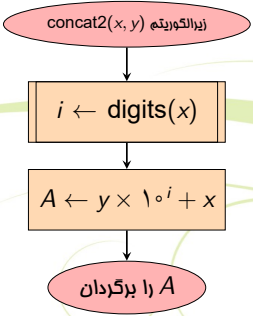
الگوریتم:

۱ **شروع** زیرالگوریتم $concat2(x, y)$

۲ $i \leftarrow \text{digits}(x)$

۳ $A \leftarrow y \times ۱۰^{i} + x$

۴ A را برگردان

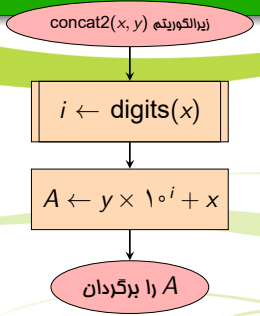


مثال ۹

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ **شروع** زیرالگوریتم $concat2(x, y)$
۲ $i \leftarrow \text{digits}(x)$
۳ $A \leftarrow y \times ۱۰^{i} + x$
۴ A را برگردان



برنامه:

```
1  long int concat2(long int x,long int y)
2  {
3     long int A;
4     int i;
5     i=digits(x);
6     A=y*pow(10,i)+x;
7     return A;
8  }
```

مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

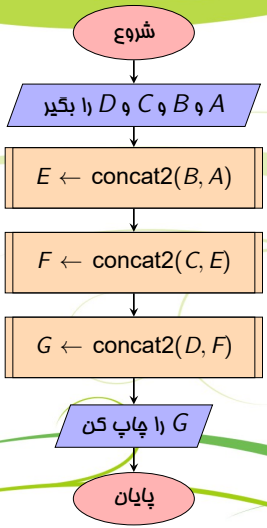
۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow \text{concat2}(B, A)$
۴ $F \leftarrow \text{concat2}(C, E)$
۵ $G \leftarrow \text{concat2}(D, F)$
۶ G را چاپ کن
۷ پایان

مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow \text{concat2}(B, A)$
۴ $F \leftarrow \text{concat2}(C, E)$
۵ $G \leftarrow \text{concat2}(D, F)$
۶ G را چاپ کن
۷ پایان



مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow \text{concat2}(B, A)$
۴ $F \leftarrow \text{concat2}(C, E)$
۵ $G \leftarrow \text{concat2}(D, F)$
۶ G را چاپ کن
۷ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  long int pow(int base,int power)
4  {
5  ....
6  }
7  int digits(long int n)
8  {
9  ....
10 }
11 long int concat2(long int x,long int y)
12 {
13 .....
14 }
15 int main()
16 {
17    long int A,B,C,D,E,F,G;
18    cout << "Give me four numbers: ";
19    cin >> A >> B >> C >> D;
20    E=concat2(B,A);
21    F=concat2(C,E);
22    G=concat2(D,F);
23    cout << "The result= " << G;
24    return 0;
25 }
```

آرگومان در مقابل پارامتر

آرگومان در مقابل پارامتر:

- در کار با زیرالگوریتم‌ها با دو کلمه آرگومان و پارامتر مواجه شدیم که معنی مشابهی داشتند. جهت جلوگیری از ایجاد ابهام، در اینجا به ذکر تعریف دقیق این دو عبارت می‌پردازیم.
- زیرالگوریتم بر مبنای یک سری داده‌های فرضی که در متغیرهایی ذخیره شده‌اند نوشته می‌شود. این متغیرها را پارامترهای زیرالگوریتم می‌نامیم.
- اما آرگومان چیست؟ آرگومان در واقع، موقع استفاده یا فراخوانی زیرالگوریتم خود را نشان می‌دهد.

آرگومان در مقابل پارامتر

آرگومان در مقابل پارامتر:

- در کار با زیرالگوریتم‌ها با دو کلمه آرگومان و پارامتر مواجه شدیم که معنی مشابهی داشتند. جهت جلوگیری از ایجاد ابهام، در اینجا به ذکر تعریف دقیق این دو عبارت می‌پردازیم.
- زیرالگوریتم بر مبنای یک سری داده‌های فرضی که در متغیرهایی ذخیره شده‌اند نوشته می‌شود. این متغیرها را پارامترهای زیرالگوریتم می‌نامیم.
- اما آرگومان چیست؟ آرگومان در واقع، موقع استفاده یا فراخوانی زیرالگوریتم خود را نشان می‌دهد.

آرگومان در مقابل پارامتر

آرگومان در مقابل پارامتر:

- در کار با زیرالگوریتم‌ها با دو کلمه آرگومان و پارامتر مواجه شدیم که معنی مشابهی داشتند. جهت جلوگیری از ایجاد ابهام، در اینجا به ذکر تعریف دقیق این دو عبارت می‌پردازیم.
- زیرالگوریتم بر مبنای یک سری داده‌های فرضی که در متغیرهایی ذخیره شده‌اند نوشته می‌شود. این متغیرها را پارامترهای زیرالگوریتم می‌نامیم.
- اما آرگومان چیست؟ آرگومان در واقع، موقع استفاده یا فراخوانی زیرالگوریتم خود را نشان می‌دهد.

با آرزوی موفقیت و بهروزی

mfarshi@yazd.ac.ir