برنامه نویسی پیمانه ای :

در ابتدای این فصل به الگوریتم و نحوه نوشتن و طرح راه حلهای رایانه ای برای حل مسائل اشاره شد ولی در هنگام مواجهه با حل مسایل به این روشها دیده می شود که با افزایش حجم مساله سختی کار بصورت نمایی زیاد می شود و عملاً نوشتن یک الگوریتم جزیی برای کل حل مساله غیر ممکن می شود.

در اینجاست که نیاز به تدوین روش برای مقابله با این مشکل احساس می شود. البته ناگفته نماند که مشابه این مشکل وقتی پیش می آید که چند نفر و بصورت مشترک بخواهند یک راه حل برای مساله طرح کنند تا روند حل مساله ساده تر شود.

روش برنامه نویسی " پیمانه ای [1]" به این صورت است که کل حل مساله به بخشهای کوچکتر –که درعین حال خود مساله های دیگری هستند که باید بصورت جزیی حل می شوند – تقسیم می شود هر بخش با صورت مساله خود (که کوچکتر از مساله اصلی است) جداگانه حل می شود و در نهایت این حلها با هم ادغام شده و کل حل مساله را تشکیل می دهند.

اگر حل کل مساله را برنامه بنامیم هر بخش کوچکتر دارای حلی است که به آن " زیر برنامه[2]" گفته می شود – در بخش "6-2-1-2" و فلوچارت توضیح دادیم که زیر روال ها (همان زیر برنامه ها) در فلوچارت کل (ادغام زیر مساله ها) با نماد مشخص می شود.

به روند تقسیم کردن مسایل بزرگ به چندین زیر مساله و حل جداگانه هر کدام و ادغام این زیر برنامه ها برنامه نویسی " پیمانه ای " گویند.

حال که اصل و مبدا این روش گفته شد می توان به مزایا و معایب استفاده از این روش پرداخت مسلماً یکی از مزایای این روش ممکن شدن پیاده سازی بسیاری از راه حلهاست که بصورت یک الگوریتم کلی آنقدر

دشوارند که عملاً غیر ممکن هستند!

برنامه نویسی به روش ″ پیمانه ای ″ علاوه بر مورد گفته شده دارای مزایای فرعی دیگری نیز هست بسیاری از اوقات مسایلی هستند که اگر در صورت آنها تغییراتی ایجاد کنیم راه حل عوض نمی شود بلکه ورودیهای راه حل عوض می شود. مثلاً انتگرال معین یک تابع خاص را در نظر بگیرید که در حل یک مساله چندین بار ولی با حدود متفاوت $\left(\int_c^d , \int_a^b\right)$ استفاده شده است.

در چنین مواردی با استفاده از روش پیمانه ای می توان فقط یکبار انتگرال را حل کرد و سپس با قرار دادن حدود متفاوت از یک بار حل در چند بخش حل کلی استفاده کرد. این امر هم کار حل مسایل کلی را ساده تر می کند و هم حجم فضای مورد نیاز برنامه را کاهش می دهد.

بسیاری اوقات راه حلهای بهینه برای مسایل بدست می آید اگر هر زیر بخش حل مساله بهینه باشد. یعنی با فرض طرح زیر برنامه هایی با بهترین مشخصات ممکن می توان بهترین برنامه را نوشت. هر چند بهینه کردن حداکثر موارد ممکن نیست ولی تقریبی از بهینه را می توان در نظر گرفت. این واقعیت منجر می شود به این که روش طراحی یک برنامه کلی نوشته شود و در آن حداکثر تلاش برای بهتر نوشتن کل حل انجام گیرد سپس هر گاه راه حلهای بهتری برای زیر مساله های خاص یافتیم آنها را جایگزین می کنیم و برنامه را بهبود می بخشیم. به عنوان مثال برنامه اصلاح کیفیت یک ″ عکس ″ را در نظر بگیرید که نیاز به یک سری محاسبه انتگرالی دارد. ما اصل راه حل را بر فرض آنکه زیر برنامه محاسبات مذکور به درستی نوشته شده طرح می کنیم و سپس زیر مساله محاسبه انتگرال را حل می کنیم. هر گاه روش جدیدی برای محاسبه انتگرالها یافتیم بهتر بود کافیست همین بخش جایگزین گردد تا کل برنامه بهتر شود. به عبارت دیگر بهتر کردن برنامه امکان پذیرتر! است.

و بالاخره در مورد این روش باید گفت وقتی به صورت پیمانه ای می اندیشید این امکان بوجود می آید که بر حسب انتخاب کاربر انواع متفاوتی از حل مساله اتفاق بیفتد. برای روشن تر شدن مطلب یک مثال عملی می زنیم:

یک مساله مهم در امر پردازش بدست آوردن مولفه های فرکانسی موج ورودی است. برای این کار دو روش وجود دارد یکی محاسبه مستقیم DFT **و دیگری استفاده از الگوریتم** FFT**. بر حسب اینکه چند مولفه فرکانسی از موج ورودی را نیاز داریم این دو روش انتخاب می شوند یعنی بهتر بودن این دو الگوریتم وابسته به تعداد المانهای فرکانسی مورد نیاز است. با استفاده از روش پیمانه ای می توان به سادگی برنامه را نوشت و فقط در مورد هر صورت مساله با توجه به تعداد المانهای مورد نیاز از زیر برنامه مناسب استفاده کرد. البته شاید این مثال خیلی آشکار نباشد و لذا برای توضیح بیشتر تا فصل 6-5-1 که در مورد زیر برنامه ها بصورت اختصاصی** *Delphi* **سخن خواهیم گفت صبر کنید.**

پیمانه اندیشیدن علاوه بر برنامه نویسی رایانه کمک شایان توجهی به انجام کارهای خود ما خواهد کرد. مشابه الگوریتم که علاوه بر کاربرد در دنیای رایانه به کارکرد ما هم در کارهای بزرگ کمک می کند پیمانه ای اندیشیدن به معنای تقسیم کار به بخشهای کوچکتر هم بسیاری از کارهای سخت را برای ما ممکن می نماید.

Modular [1]

Subprogram [2]