فهرست

فصل اول: مقدمه ای بر ساختمان داده ها

۱-۱- مقدمه

کامپیوتر وسیله ای است که داده ها و الگوریتم ها را گرفته، الگوریتم ها را بر روی داده ها اجرا می کند. داده ها و الگوریتم ها دو پارامتری هستند که در نوشتن برنامه های خوب موثرند. کارایی برنامه وقتی تضمین می شود که داده ها را خوب سازمان دهی کرده الگوریتم خوبی برای پردازش و تجزیه و تحلیل آنها انتخاب کنیم. بنابراین لازم است دانشجویان کامپیوتر با سازمان دهی دادهها آشنا بوده الگوریتم ها را به خوبی بشناسند.

پردازش داده های کاربر(داده هایی که در دنیای واقعی وجود دارند)، مستلزم کارهای زیر است:

۱- نمایش حافظه داده های کاربر: داده های کاربر باید طوری ذخیره شوند که کامپیوتر آن را درک کند.

۲- بازیابی داده های ذخیره شده: داده های ذخیره شده در کامپیوتر باید طوری بازیابی شوند که کاربر آنها را درک کند.

۳- تبدیل داده های کاربر: اعمال مختلفی که لازم است بر روی داده های کاربر اجرا شوند تا آن را از شکلی به شکل دیگر درآورند.

داده ها می توانند به صورت های مختلفی سازمان دهی شوند. مدل منطقی یا ریاضی سازمان دهی داده ها به یک شکل خاص، ساختمان داده نام دارد. به عبارت دیگر، شیوه قرار گرفتن داده ها در حافظه کامپیوتر یا دیسک، ساختمان داده ها نامیده می شود.

۱-۲- حل مسئله به وسیله کامپیوتر

توسعه نرم افزار در چند مرحله انجام می شود که چرخه زندگی نرم افزار را تشکیل می دهد:

- **تحلیل مسئله و مشخصات:** در این مرحله مسئله تحلیل می شود و مشخصاتی برای مسئله تنظیم می گردد.
- **طراحی:** برنامه ریزی برای حل مسئله ای که مطرح شده است.
- **کدنویسی:** طراحی، به وسیله زبان های برنامه سازی پیاده سازی می شود.
- **تست، اجرا و اشکال زدایی:** برنامه امتحان می شودو خطاهای آن برطرف می گردد.
- **نگهداری:** برنامه نوسازی و اصلاح می شود تا نیازهای جدید کاربران را برآورده کند.

۱-۳- نکاتی راجع به الگوریتم ها

در علم کامپیوتر، الگوریتم مجموعه ای از دستورالعمل ها است که توسط کامپیوتر اجرا می شود و به این ترتیب محدودیت هایی به آن اعمال می گردد:

۱- الگوریتم باید بدون ابهام، معین و قطعی باشد، به طوری که هر مرحله روشن و واضح باشد.

۲- الگوریتم ها باید به قدری سازده باشند که توسط کامپیوتر قابل اجرا باشند.

۳- باید خاتمه پذیر باشند، یعنی پس از تعدادی مراحل، خاتمه یابند.

۱-۳-۱- کارایی الگوریتم

برای حل یک مسئله ممکن است الگوریتم های مختلفی وجود داشته باشند که هر کدام دارای مزایا و معایب خاص خودشان هستند. کارایی الگوریتم بر اساس دو معیار اندازه گیری می شود. یکی از آنها بهره وری از فضا[1] و دیگری کارایی زمان[2] است. بهره وری فضا، حافظه ای است که برای ذخیره داده ها لازم است و کارایی زمان، زمان لازم برای پردازش داده ها است. متاسفانه همیشه نمی توان هم فضای موردنیاز و هم زمان موردنیاز را کم کرد. الگوریتم هایی که به فضای کمتری نیاز دارند، معمولاً نسبت

[1] - space utilization
[2] - time efficiency

به آنهایی که به فضای بیشتری نیاز دارند، کندتر هستند. لذا، برنامه نویس باید بین نیازمندی های فضا و کارایی زمان، توان ایجاد کند.

۲-۳-۱- عوامل موثر در کارایی الگوریتم

منظور از کارایی الگوریتم، مدت زمان اجرای آن می باشد.

٤-۱- نکاتی در مورد کدنویسی

کدنویسی، مرحله ای از دوران زندگی نرم افزار است. تعدادی از عملیات برنامه نویسی را باید رعایت کرد تا قابلیت خوانایی و قابلیت درک کد افزایش یابد.

قاعده اول: برنامه ها و زیربرنامه ها باید ساخت یافته باشند.

قاعده دوم: تمام کدها باید مستندسازی شوند.

قاعده سوم: کد باید زیبا باشد. فرمت و تورفتگی هایی داشته باشد که خوانایی آن را بالا ببرد.

٥-۱- نوع داده انتزاعی

وقتی در برنامه ای به نوعی داده نیاز باشد که در آن زبان وجود ندارد، برنامه نویس باید نوع موردنظرش را ایجاد کند. بنابراین، باید چگونگی ذخیره داده ها و عملیاتی را مشخص کند که بر روی آن داده ها عمل می کنند. نوع داده ای را که برنامه نویس ایجاد می کند، نوع داده انتزاعی گویند.

نوع داده انتزاعی [1](ADT) یک مدل ریاضی است که عملیاتی بر روی آن مدل تعریف شده اند.

٦-۱- آرایه ها

آرایه ساده ترین ساختمان داده ای است که در زبان C وجود دارد. ساده ترین شکل آن، آرایه یک بعدی است که بردار نیز نامیده می شود. آرایه یک بعدی، مجموعه مرتب و محدودی از عناصر همگن است. منظور از «محدود» این است که تعداد عناصر آرایه مشخص است. این تعداد ممکن است کوچک یا بزرگ باشد. منظور از «مرتب» این است که عناصر آرایه دارای ترتیب خاصی اند: عنصر صفر، عنصر یکم، عنصر دوم و غیره. منظور از «همگن» این است که کلیه عناصر آرایه باید همنوع باشند.

۷-۱- آرایه های یک بعدی

آرایه یک بعدی که نام دیگر آن بردار است، برای ذخیره مجموعه ای از عناصر همنوع به کار می رود.

۱-۷-۱- پیاده سازی آرایه یک بعدی

آرایه های یک بعدی را به آسانی می توان پیاده سازی کرد. دستور زیر را در نظر بگیرید:

```
Int a[10];
```

این دستور ۱۰ محل متوالی حافظه را تخصیص می دهد که در هر محل می توان یک مقدار صحیح را ذخیره کرد.آدرس اولین محل، آدرس پایه نام دارد و با base(a) مشخص می شود. اگر فرض کنید هر مقدار صحیح چهار بایت از فضای حافظه را اشغال می کند، آنگاه اولین عنصر آرایه با شروع از آدرس base(a) در چهار بایت از حافظه ذخیره می شود. عنصر a[1] با شروع از آدرس base(a)+4 در چهار بایت ذخیره می شود.

۲-۷-۱- کاربردهای آرایه های یک بعدی

آرایه های یک بعدی برای ذخیره داده های زیادی به کار می روند و همه مقادیر موجود در آرایه به روش یکسانی دستیابی می شوند. یکی از کاربردهای آرایه در الگوریتم مرتب سازی و جست و جو است.

[1] - Abstract Data Type

جست و جو در آرایه

یکی دیگر از اعمالی که در آرایه های یک بعدی صورت می گیرد، جست و جوی مقداری در یک آرایه است. جست و جو می تواند به صورت ترتیبی یا دو دویی انجام شود. جست و جوی دودویی در آرایه مرتب صورت می گیرد.

۸-۱- آرایه های دو بعدی

آرایه های دو بعدی در C به صورت زیر اعلان می شوند:

اعلان آرایه دو بعدی
شکل کلی: نوع name[DIM_1] [DIM_2] نوع name[DIM_1] [DIM_2] = {مقادیر اولیه}; DIM_1 , DIM_2 به ترتیب اندیس های سطر و ستون هستند. در حالت دوم مقادیر اولیه مشخص می کنند که عناصر آرایه چه مقادیری داشته باشند. هدف: به تعداد $DIM_1 \times DIM_2$ محل متوالی حافظه به ارایه تخصیص می یابد.

آرایه دو بعدی یک ساختمان داده منطقی است که در برنامه نویسی و حل مسائل به کار گرفته می شود. به خصوص برای توصیف اشیایی که به طور فیزیکی دو بعدی اند، مثل نقشه یا صفحه شطرنج مفید هستند. به عنوان مثال، برای ذخیره نمرات ۳۰ دانشجو که هر دانشجو ۵ بار امتحان داده باشد، از آرایه دو بعدی به صورت زیر استفاده می کنیم:

```
Float grage[30][15];
```

۱-۹-۱- کاربرد آرایه های دو بعدی

نام دیگر آرایه های دو بعدی، ماتریس است که کاربردهای فراوانی در حل مسئله ها دارد. نمونه هایی از ماتریس ها در زیر مشاهده می شوند که ماتریس a را ۳×۴ می گوییم و به صورت $a_{۳\times۴}$ می نویسیم و ماتریس b را ۲×۵ می گوییم و به صورت $b_{۲\times۵}$ می نویسیم:

$$a_{3\times4}\begin{bmatrix}-2 & 3 & 1 & 4\\ 5 & 1 & 7 & 9\\ 3 & 2 & 1 & 5\end{bmatrix} \quad b_{2\times5}\begin{bmatrix}1 & 4 & 3 & 2 & 5\\ 2 & 7 & 8 & 1 & 9\end{bmatrix}$$

۲-۹-۱- ماتریس های اسپارس

بعضی از ماتریس ها وجود دارند که تعداد زیادی از عناصر آنها صفر است، به عنوان مثال، ممکن است در مسئله ای به ماتریسی به ابعاد ۱۰۰۰×۱۰۰۰ نیاز داشته باشیم که حاوی یک میلیون عنصر است. از بین این عناصر ممکن است ۱۰۰۰ عنصر صفر باشند. ماتریسی که تعداد زیادی از عناصر آن صفر باشند، ماتریس اسپارس[1] نامیده می شود.

ترانهاده ماتریس اسپارس

منظور از ترانهاده ماتریس، ماتریس دیگری است که جای سطر و ستون های آن عوض شده باشد. طریقه به دست آوردن ترانهاده ماتریس معمولی را مشاهده کردید.

[1] - sparse

برای این که راه حلی برای به دست آوردن ترانهاده ماتریس اسپارس پیدا کنیم، ترانهاده ماتریس اسپارس زیر را به دست می آوریم:

	4	5	4
	0	2	5
B=	1	1	7
	2	0	5
	2	1	6

۱۰-۱- آرایه های چند بعدی

در C می توان از آرایه هایی با بیش از دو بعد نیز استفاده کرد. به عنوان مثال، آرایه سه بعدی را می توان به صورت زیر تعریف کرد:

```
Int b[3][2][4];
```

۱۱-۱- مشکلات آرایه

آرایه ها چند مشکل دارند که در هنگام استفاده از آنها باید در نظر گرفته شوند:

- حد و مرز آرایه ها در C کنترل نمی شود. یعنی هنگامی که از اندیس برای دستیابی به عنصری از آرایه استفاده می گردد، کنترل نمی شود که آیا این اندیس در بازه ای که هنگام اعلان آرایه مشخص شده است قرار دارد یا خیر.
- ظرفیت آرایه در طول اجرای برنامه تغییر نمی کند. معنایش این است که ظرفیت آرایه باید در هنگام اعلان آرایه به اندازه کافی بزرگ تعیین شود تا همه داده های ممکن را ذخیره کند. این کار ممکن است منجر به اتلاف حافظه شود. به عنوان مثال، ذخیره ۲۵ عنصر در آرایه ای به ظرفیت ۵۰۰۰ بایت موجب می شود تا حافظه زیادی به هدر برود.
- درج کردن مقداری در بین مقادیر دیگر در آرایه، مستلزم جابه جایی عناصر است.
- حذف مقداری از آرایه نیز مستلزم جابه جایی عناصر است.

۱۲-۱- نشانه گذاری O بزرگ

نماد O را می توان اختصار واژه های "Order of magnitude" دانست. این نماد را نماد O بزرگ یا Big-O می نامند.

ساده ترین روش محاسبه ی O مربوط به الگوریتم یا برنامه، در نظر گرفتن حلقه های تکرار و تو در تو بودن آنها است. اگر فرض کنیم بدنه حلقه فقط حاوی دستورات ساده است. یک حلقه از $O(n)$، حلقه تو در تو از $O(n^2)$ و حلقه ای با درجه ی تو در تویی ۳ از $O(n^3)$ است و غیره. علاوه بر این، باید تعداد دفعات تکرار اجرای حلقه را تعیین کنید.

فصل دوم: پشته ها

پشته ساختمان داده ای است که عمل حذف و اضافه کردن عناصر آن، به نام بالای پشته انجام می شود. به همین دلیل آن را ساختمان داده LIFO[1] می نامند که به معنای خروج به ترتیب عکس ورود است. یعنی عنصری که دیرتر از همه وارد پشته شد، زودتر از همه از پشته خارج می شود. به چند نمونه از پشته توجه کنید: وقتی تعدادی کتاب را روی هم می چینید، پشته ای از کتاب ها را ایجاد می کنید. هنگام برداشتن کتاب ها، آخرین کتابی را که گذاشتید، زودتر از سایر کتاب ها برمی دارید.

پیاده سازی پشته با آرایه

در این روش، برای ذخیره عناصر پشته از آرایه استفاده می کنیم و برای مدل سازی این نوع پشته از متغیری به نام myTop برای نگهداری بالای پشته استفاده خواهیم کرد. عمل اضافه کردن و حذف عنصر از پشته، از محل myTop انجام می شود. بنابراین، داده های موردنیاز برای پشته عبارتند از:

- آرایه ای که عناصر پشته را نگهداری می کند.
- یک متغیر صحیح که بالای پشته را مشخص نماید.

پیاده سازی عمل ایجاد پشته

عمل ایجاد پشته، باید یک آرایه و یک متغیر نشان دهنده بالای پشته را تعریف کند و پشته خالی را ایجاد نماید. متغیر نشان دهنده بالای پشته، MyTop است، که در پشته خالی، برابر با ۱- است. بنابراین، عمل ایجاد پشته به صورت زیر پیاده سازی می شود:

```
#define SIZE 100
Struct stack {
        Int myTop;
        Int items[SIZE];
};
Stack s;
s.myTop = -1;
```

پیاده سازی عمل بازیابی از پشته

عمل بازیابی از پشته، عنصر بالای پشته را بازیابی می کند، ولی آن را از پشته حذف نمی کند. بدیهی است که این عمل باید خالی بودن پشته را بررسی کند. اگر پشته خالی باشد، امکان بازیابی عنصر وجود ندارد. این تابع را می توان به صورت زیر نوشت:

```
Int top(stack*s)
{
        If(empty(s))
        {
                Printf("Stack underflow.");
                Exit(1);
        }
        Else
                Return(s-> items[s-> myTop]);
}
```

کاربرد پشته ها در فراخوانی توابع

هر وقت تابعی فراخوانی می شود، یک رکورد فعالیت[2] برای آن تابع ایجاد می گردد و محیط فعلی را برای آن تابع ذخیره می کند. رکورد فعالیت شامل اطلاعات زیر است:

- پارامترها

[1] - Last in First Out
[2] - activation record

- اطلاعات حالت فراخوان، مثل محتویات ثبات ها و آدرس های برگشت
- متغیرهای محلی
- حافظه های موقت برای انجام محاسباتی میانی

چون ممکن است هر تابع، توابع دیگری را فراخوانی کند و اجرای خود آن تابع به تعویق افتد، رکورد فعالیت آن باید طوری ذخیره شود که وقتی تابع از سرگرفته می شود، بتوان به رکورد فعالیت آن دست یافت.

ساختمان داده ای که رکوردهای فعالیت در آن ذخیره می شوند، باید رفتار LIFO(خروج به ترتیب عکس ورود) داشته باشد، زیرا اولین تابعی که خاتمه می یابد، آخرین تابعی است که فراخوانی شده است و رکورد فعالیت آن باید زودتر از همه بازیابی شود. لذا، پشته ساختمان داده مناسبی برای این کار است. چون این پشته در زمان اجرا دستکاری می شود، پشته زمان اجرا نام دارد.

تبدیل عبارات میانوند به پسوند

همان طور که می دانید ارزیابی عبارتی مثل A+B*C که به صورت میانوند نوشته شده است، مستلزم اطلاع از تقدم عملگرهای + و * است. می دانید که تقدم * از + بیشتر است. بنابراین، عبارت A+B*C را می توان به صورت A+(B*C) نوشت. اکنون فرض کنید می خواهیم عبارت میانوند A+B*C را به عبارت پسوندی تبدیل کنیم. تنها روش تبدیل یک عبارت میانوند به عبارت پسوند این است که ابتدا قسمت های با تقدم بالا به عبارت پسوند تبدیل شوندو عبارت حاصل به عنوان یک عملوند محسوب شده با سایر عملوندها به عبارت پسوند تبدیل شود. بنابراین، عبارت A+B*C به صورت زیر به عبارت پسوند تبدیل می گردد:

A+B*C عبارت میانوند

A+(BC*) تبدیل عمل ضرب

A(BC*)+ تبدیل عمل جمع

ABC*+ عبارت پسوندی

الگوریتم تبدیل عبارت میانوندی به پسوندی

دریافتی: عبارت میانوند

برگشتی: عبارت پسوندی

۱- پشته ای خالی برای عملگرها ایجاد کنید.

۲- تا زمانی که خطایی رخ نداده است و به انتهای عبارت میاوند نرسیدی اعمال زیر را انجام بده:

الف) نشانه بعدی(ثابت، متغیر، عملگر، پرانتز باز، پرانتز بسته) را از عبارت میاوند دریافت کن.

ب) اگر نشانه

- **پرانتز باز** است، آن را در پشته قرار بده.
- **پرانتز بسته** است، عناصر پشته را آنقدر حذف کنید و در عبارت پسوندی قرار دهید تا به پرانتز باز برسید، ولی پرانتز باز را در عبارت پسوندی قرار ندهید(اگر پشته خالی شد و به پرانتز باز نرسیدید، خطایی در عبارت میاوند وجود دارد)
- **عملگر** است، اگر پشته خالی است، یا تقدم این عملگر از تقدم عملگر بالای پشته بیشتر است، آن را در پشته قرار دهید، وگرنه عنصر بالای پشته را حذف کنید و در عبارت پسوندی قرار دهید. این کار را برای عملگر جدید موجود در بالای پشته تکرار کنید.

توجه: تقدم پرانتز بازی که در پشته قرار دارد، از تقدم سایر عملگرها کمتر است.

- **عملوند** است، آن را در عبارت پسوندی قرار دهید.

۳- وقتی به انتهای عبارت میاوند رسیدی، عناصر موجود در پشته را حذف و در عبارت پسوندی قرار دهید تا پشته خالی شود.