پشته (stack) :

پشته یک لیست مرتب شده است که عملیات درج و حذف از یک طرف آن که تاپ(top) نامیده می شود انجام می شود . (آخرین عنصری که به پشته وارد شده اولین عنصری است که از پشته خارج می شود .)

ساده ترین روش برای پیاده سازی پشته استفاده از آرایه است . روش دیگر استفاده از لیست پیوندی است . در این حالت نیاز به یک آرایه برای ذخیره کردن عناصر و یک اشاره گر به آرایه به نام top است. وقتی که پشته خالی باشد top مقدار 1- را نشان می دهد. در شکل زیر یک پشته را مشاهده می کنید که به ترتیب عناصر A ، B و C در آن درج شده اند و top به آخرین عنصر وارد شده اشاره میکند.

	4	
	3	
top à	2	C
	1	B
	0	A

نمایی از یک پشته

عملیات درج در پشته push و عملیات حذف از پشته pop نام دارد . کلاس پشته به صورت زیر پیاده سازی می شود. ما فرض کردیم که عناصر پشته از نوع float باشد و با توجه به اینکه برای ساختن پشته از آرایه استفاده کرده ایم، مجبور شده ایم تا تعداد عناصر آن را مشخص نماییم.

```
#define MAX_STACK_SIZE 100
class stack {
      private:
              float element[MAX_STACK_SIZE];
              int top;
      public:
            stack() { init(); };
            void  init(); { top=-1; };
            int        isEmpty();
            int        isFull();
            void  push(float x);
            float      pop();
};
```

در این قسمت به پیاده سازی هر یک از توابع عضو پشته می پردازیم.

1- تابع عضو isFull

این تابع تست پر بودن پشته را انجام می دهد. اگر پشته پر باشد خروجی این تابع یک است و اگر پشته پر نباشد خروجی این تابع صفر است. پشته وقتی پر است که top به آخرین عنصر آرایه اشاره نماید.

```
int stack::isFull(){
      return top==MAX_STACK_SIZE-1;
}
```

2- تابع عضو isEmpty

این تابع تست خالی بودن پشته را انجام می دهد. اگر پشته خالی باشد خروجی این تابع یک است و اگر پشته خالی نباشد خروجی این تابع صفر است. پشته وقتی خالی است که top برابر عدد -1 باشد.

```
int stack::isEmpty() {
      return (top==-1);
}
```

3- عملیات push

Push یک عنصر را به پشته اضافه می کند. وقتی که عملیات push برای درج یک عنصر در پشته انجام میشود، اگر پشته پر نباشد، ابتدا به top یک واحد اضافه می شود وسپس آن عنصر در محلی که top نشان می دهد قرار می گیرد .

```
void  stack::push(float x){
      if (isFull()) {
            printf("Error:Stack is Full");
            exit(1);
      }
      top++;
      element[top]=x;
}
```

4-عملیات pop

Pop یک عنصر را از بالای پشته حذف می کند. Pop به شرطی می تواند عملیات حذف را انجام دهد که پشته خالی نباشد. بنابراین قبل از برداشتن عنصر از بالای پشته، خالی بودن پشته را بررسی می کند.

اگر پشته خالی نباشد، عنصر بالای پشته در یک متغییر کمکی ذخیره می‌شود و سپس از مقدار top یک واحد کم می شود .

```
float stack::pop() {
      if (isEmpty()){
            printf("Error : Stack is Empty ");
            exit(1);
      }
      float  x= element[top];
      top--;
      return x;
}
```

کاربرد پشته ها:

- توابع بازگشتی
- ارزشیابی عبارات محاسباتی
- الگوریتم های مسیر و حرکت
- پیمایش عمقی درخت و گراف.

به عنوان مثال از کاربرد پشته در توابع بازگشتی به توابع بازگشتی فاکتوریل و فیبوناتچی اشاره می کنیم.

تابع بازگشتی فاکتوریل:

تابع بازگشتی فاکتوریل به صورت زیر تعریف می شود.

```
int fact(int   n)
{
   if (n<=1) return  1;
   else
      return   n *fact( n-1);
}
```

				1				
			2	2	2			
		3	3	3	3	3		
	4	4	4	4	4	4	4	
5	5	5	5	5	5	5	5	5
N	5	4	3	2	1	2	3	4	5
Return value					1	2	6	24	120

دنباله فیبونانچی دنباله ای ازاعداد است به صورت زیر

1 1 2 3 5 8 13 ....

که هر جمله در این دنباله برابر به مجموع دو جمله قبلی است، به جز جمله صفرم و جمله یکم که برابر یک هستند

F0=1 ,f1=1 , fn= fn-1 + fn-2

تابع بازگشتی محاسبه F_n را بنویسید و سپس وضعیت پشته را برای fibo(4) نشان دهید.

```
int  fibo(int n)
{
   if (n<=1) return 1;
   else  retuen fibo(n-1)+fibo(n-2);
}
```

		0					0	
		1	1			1	1	1
	2	2	2	2		2	2	2	2
	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
N	4	2	0	1	2	3	1	2	0
Return value	→	→	1	1	2		1		1

	2								
	3	3							
	4	4	4						
N	1	2	3	4					
Return value	1	2	3	5					

ارزشیابی عبارات محاسباتی:

عبارتهای محاسباتی به سه صورت قابل نمایش است :

1)روش نمایش ،infix یا میانوندی: در این روش عملگر ها در بین عملوندها قرار می گیرند.

2)روش نمایش یا post fix ،یا پسوندی: در این روش عملگرها بعد از عملوندها می آید.

3) روش نمایش prefix، یا پیشوندی: در این روش عملگر قبل از عملوندها می آید

به طور مثال عبارت a+b را می توان در هر یک از سه روش فوق به صورت جدول زیر نشان داد

infix	a+b
Postfix	a b +
prefix	+ a b

نکته: 1. در روش infix برای به هم زدن اولویت عملگرها باید از پرانتز استفاده کنیم ولی در روشهای postfix و prefix نیاز به پرانتز گذاری عبارت نیست.

ارزیابی یک عبارت postfix :

به عنوان یک مثال دیگر از کاربرد پشته ها می توان به استفاده از آن در ارزیابی عبارت postfix اشاره نمود .لگوریتم ارزیابی یک عبارت postfix به صورت زیر می باشد.

الگوریتم: ارزیابی یک عبارت postfix

ورودی: یک عبارت postfix

خروجی: مقدار ارزیابی شده.

شروع

یک پشته از اعداد به نام s در حافظه در نظر بگیر.

پشته s را برای استفاده آماده کن.

عبارت postfix از چپ به راست مورد بررسی قرار می گیرد.

یک توکن را از عبارت postfix جدا کن.

تا زمانی که توکن علامت انتهای جمله نبود کارهای زیر را تکرار کن

اگر توکن یک عدد باشد آن را در پشته s اضافه کن.

اگر توکن یک عملگر بود آنگاه

یک عدد از پشته خارج کن و آن را در متغییر X ذخیره کن.

یک عدد از پشته خارج کن و آن را در متغییر Y ذخیره کن.

اگر توکن عملگر جمع بود آنگاه Z=Y+X.

اگر توکن عملگر تفریق بود آنگاه Z=Y-X.

اگر توکن عملگر تقسیم بود آنگاه Z=Y/X.

اگر توکن عملگر ضرب بود آنگاه Z=Y*X.

Z را به پشته s اضافه کن.

پایان اگر.

یک توکن را از عبارت postfix جدا کن.

پایان تا زمانی که.

عددی را از پشته s خارج کن و در R ذخیره کن.

R نتیجه ارزیابی است.

پایان الگوریتم.

برای بررسی نحوه عملکرد الگوریتم فوق می توانید فایل نمایشی postfix_eval را مشاهده نمایید.

ارزیابی یک عبارت infix:

برای ارزیابی یک عبارت infix که ممکن است حاوی پرانتز باشد می توانیم ابتدا عبارت postfix معادل آن را محاسبه کرد و سپس عملیات ارزیابی را با استفاده از الگوریتم ارزیابی عبارت postfix انجام داد. پس مساله اصلی در اینجا محاسبه عبارت postfix از روی عبارت infix است که روش هایی را برای آن مطرح می کنیم.

روش پرانتزگذاری:

در این روش ابتدا جمله infix را به طور کامل با توجه به اولویت عملگرها پرانتزگذاری میکنیم و سپس در سمت راست پرانتز بسته ها عملگر مربوط به آن پرانتز را می نویسیم. درنهایت جمله postfix را از روی عبارت حاصل با حذف پرانتزها

می نویسیم.

مثال:

```
Infix :   a*b+c
```

بعد از پرانتز گذاری داریم:

```
((a*b)+c)
```

بعد از نوشتن عملگرها بر روی پرانتز بسته ها داریم:

$((a\ b)^{*}\ c)^{+}$

و درنهایت داریم:

```
Postfix= a b * c +
```

مثال:

```
Infix = a*(b+c)-g/d
```

بعد از پرانتز گذاری داریم:

```
((a*(b+c))-(g/d))
```

بعد از نوشتن عملگرها روی پرانتز بسته ها داریم:

$((a\ (b\ c)^{+})^{*}\ (g\ d)^{/})^{+}$

و در نهایت داریم:

```
Postfix= a b c + * g d / +
```

استفاده از پشته:

در این روش از یک پشته برای ذخیره کردن عملگرهای ورودی استفاده می شود. در این الگوریتم باید اولویت عملگرها را نیز در نظر بگیریم برای اینکار از یک ماتریس اولویت برای مشخص کردن اولویت ها استفاده می کنیم که به صورت زیر مقدار دهی شده است .

ops	?	+	-	*	/	(
?	0	0	0	0	0	0
+	1	0	0	-1	-1	1
-	1	0	0	-1	-1	1
*	1	**1**	1	0	0	1
/	1	1	1	0	0	1
(	1	1	1	1	1	1

ماتریس فوق اولویت عملگر ورودی را نسبت به عملگرهای درون پشته نشان می دهد. عدد یک نشان دهنده بالا بودن اولویت، عدد صفر نشان دهنده برابر بودن اولویت و عدد منفی یک نشان دهنده پایین تر بودن اولویت است. به طور مثال اگر عملگر ورودی *(ضرب) باشد و عملگر بالای پشته +(جمع) باشد در آنصورت با توجه به ماتریس اولویت ضرب جمع بیشتر است(ماتریس بالا را ببینید). نکته مود توجه در اولویت ها این است که اگر پرانتز باز یک عملگر ورودی باشد اولویت آن از همه عملگرهای درون پشته بیشتر است و اگر پرانتز باز در داخل پشته باشد آنگاه اولویت آن از همه عملگرهای ورودی کمتر است.

الگوریتم تبدبل عبارت infix به postfix به صورت زیر بان می شود:

الگوریتم: تبدیل عبارت infix به postfix

ورودی: عبارت infix

خروجی: عبارت postfix

شروع

یک پشته از عملگرها به نام s در حافظه در نظر بگیر.

پشته s را برای استفاده آماده کن.

عبارت infix از چپ به راست مورد بررسی قرار می گیرد.

یک توکن از عبارت infix جدا کن.

تا وقتیکه که توکن علامت انتهای جمله نبود آنگاه کارهای زیر را انجام بده

اگر توکن یک عملگر است آنگاه

اگر اولویت توکن از عملگر بالای پشته s بیشتر بود آنگاه

توکن را به پشته s اضافه کن

در غیر اینصورت

تا زمانیکه اولویت توکن از عملگر بالای پشته کمتر یا مساوی

بود

یک عملگر از بالای پشته s خارج کن.

عملگر خارج شده را در خروجی جمله postfix

بنویس.

پایان تا وقتیکه

پایان اگر

پایان اگر در غیر اینصورت

اگر توکن علامت پرانتز بسته بود

تا زمانیکه عملگر بالای پشته پرانتز بسته نبود

یک عملگر از بالای پشته s خارج کن.

عملگر خارج شده را در خروجی جمله postfix بنویس.

پایان تا وقتیکه

عملگر پرانتز بسته را از پشته s خارج کن

پایان اگر در غیر اینصورت

توکن را در خروجی جمله postfix بنویس

پایان تاوقتیکه

تا وقتیکه پشته s خالی نبود

یک عملگر از بالای پشته s خارج کن.

عملگر خارج شده را در خروجی جمله postfix بنویس.

پایان تاوقتیکه

پایان الگوریتم