پشته

پشته لیست خطی است که تنها از یک انتهای آن می توان به عناصر دسترسی پیدا کرد. پشته برای نگهداری موقت داده ها در بسیاری از نرم افزارها کاربرد دارد.

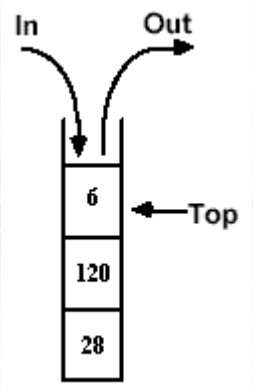
تعریف

پشته لیست خطی است که عملیات حذف و اضافه تنها از یك طرف آن به نام Top صورت می گیرد. در هر لحظه فقط عنصر بالائی پشته قابل دسترس است .

آخرین عنصری که به پشته اضافه می شود اولین عنصری است که از آن حذف می شود. یعنی عناصر عکس ترتیب ورود از پشته خارج می شود به همین دلیل پشته یک ساختمان داده LIFO (last in, first out) محسوب می شود .

می توان پشته را ساختمان داده FILO (first in, last out) هم نامید زیرا اولین داده ای که در پشته ذخیره می شود آخرین داده ای است که خارج می شود .

عمل اضافه کردن عنصر جدیدی در پشته push و عمل حذف عنصری از پشته pop نامیده می شود. عملیات دیگری نظیر خواندن عنصر بالای پشته یا بررسی خالی بودن پشته نیز ممکن است در برنامه های مختلف به کاربیاید .



نمایش پشته

پیاده‌سازی پشته با آرایه

ساده‌ترین روش پیاده‌سازی پشته استفاده از یك آرایه یك بعدی و یک متغیر Top است .

```
const int MaxSize = 100;
ItemType Stack[MaxSize];
int Top;
```

Topاندیس عنصر بالای پشته Stack را نگه می دارد. Top=0 دلالت بر خالی بودن پشته دارد .

برای Push کردن یک داده جدید به پشته، Top یکی اضافه می شود و داده جدید در آرایه در اندیس Top درج می کند .

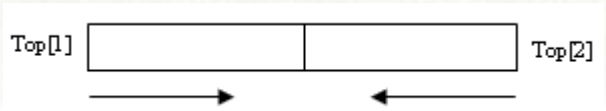
```
void Push( const ItemType &Item)
{
if (Top == MaxSize){
  cerr << "ERROR: Cannot insert -- Stack is full" << endl;
  exit(1);
  }
else{
  Top++;
  Stack[top] = Item;
  }
}
```

برای Pop کردن از پشته، داده ای که در اندیس Top قرار دارد برگردانده می شود و Top یک واحد کم می شود .

```
void Pop(ItemType & Item)
{
if (IsEmpty()){
  cerr << "ERROR: Cannot remove -- Stack is empty" << endl;
  exit(1);
  }
else{
  Item = Stack[Top];
  top--;
  }
}
```

پشته چندگانه

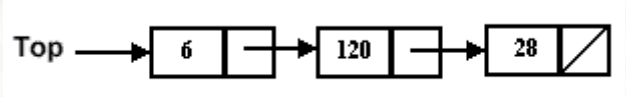
اگر نیاز به دو پشته در برنامه باشد از یك آرایه یك بعدی استفاده می شود که Top[1] ابتدای پشته اول و Top[2] ابتدای پشته دوم است. پشته ها به سمت همدیگر رشد می‌کنند .



وقتی به n پشته احتیاج داریم آرایه به n قسمت تقسیم می شود که هر قسمت به یک پشته اختصاص داده می شود. برای هر پشته b[i] پایین ترین مکان پشته و t[i] بالای پشته i را نشان می دهد. اگر t[i]=b[i] باشد یعنی پشته i خالی است و اگر t[i]=b[i+1] یعنی پشته i ام پر شده است.

پیاده سازی پشته با لیست پیوندی

چون پشته دنباله ای از عناصر داده ای است می توان آنرا در لیست پیوندی ذخیره کرد. اعمال درج و حذف از ابتدای لیست صورت می گیرند. Top آدرس اولین عنصر لیست را نگه می دارد .



کاربرهای پشته

پشته در حل مسائل مختلفی کاربرد دارد که از جمله می توان به موارد زیر اشاره کرد :

- معکوس کردن ترتیب عناصر لیست
- ذخیره آدرس‌های برگشتي و متغیرهای محلی در توابع
- تبدیل و ارزیابی عبارات
- برج هانوی
- مسیر پرپیچ و خم

معکوس کردن ترتیب عناصر لیست

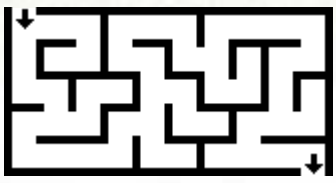
چون عناصر عکس ترتیب ورود از پشته خارج می شود پشته روش مناسبی برای معکوس کردن ترتیب عناصر یک لیست است. برای این کار کافی است ابتدا کلیه عناصر به ترتیب در پشته Push شوند سپس یکی یکی از پشته حذف شوند. لیست خروجی حاصل عکس لیست اولیه است .

مثال. الگوریتم زیر ترتیب عناصر آرایه List با n عنصر را عکس می کند.

```
for (i:=1 to n)
  Push(List[i])
end for
for (i:=1 to n)
  Pop(List[i])
end for
```

مسیر پرپیچ و خم

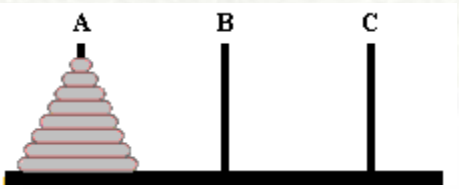
ماز (maze) مسیر پرپیچ و خمی است که حل کننده مسئله باید راهی را به بیرون پیدا کند. مسیر و موانع در ماز از قبل تعریف شده است. یکی از ساده ترین روش ها برای پیاده سازی ماز استفاده از کامپیوتر است؛ توسط يك آرایه دوبعدی n×m که صفر بودن سلولی در آن به معنی بازبودن مسیر و یک بودن به معنی مانع است .



روش حل ماز به صورت سعی و خطا است. یک مسیر تصادفی انتخاب می شود اگر راه اشتباهی بود و در مسیر بسته گیرکردیم به عقب برمی گردیم و مسیر دیگری که قبلا طی نشده را امتحان می کنیم. برای برگشت به موقعیت قبلی نیاز است قبل از رفتن به موقعیت جدید موقعیت فعلی در پشته ذخیره شود .

برج هانوی

معمای برج هانوی (Tower of Hanoi) توسط ریاضیدان فرانسوی Edouard Lucas در سال ۱۸۸۳ اختراع شد. سه میلهA ، Bو C را درنظربگیرید که روی میله A تعداد n دیسک با اندازه های مختلف از بزرگ به کوچک روی هم قرار داده شده اند. هدف بازی برج هانوی این است که دیسک ها با استفاده از میله کمکی B به میله C منتقل شوند .



قوانین بازی به صورت زیر است:

- درهربار تنها یک دیسک را می توان انتقال داد
- تنها دیسک بالائی هر میله را می توان به میله دیگر منتقل کرد.
- درهیچ زمانی نمی توان دیسک بزرگ تر را روی دیسک کوچک تر قرار داد

مثال. اگر تعداد دیسک ها ۳ باشد. ترتیب جابه جائی دیسک ها به صورت زیر می شود.

A→C A→B C→B A→C B→A B→C A→C

پیاده سازی با توابع بازگشتی

```
Tower ( n,A,B,C)
If (n=0)
  exit
else
  Tower(n-1,A,C,B)
  A->C
  Tower(n-1,B,A,C)
end if
```