لیست پیوندی

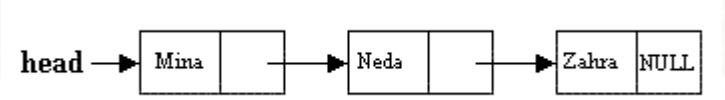
لیست پیوندی ساختاری است که ترتیب خطی عناصر داده ای در آن توسط اشاره گرها تعیین می شود.

لیست پیوندی یک طرفه

یک لیست پیوندی یک طرفه (Singly-linked list) دنباله ای از عناصر داده ای به نام گره (node)است که ترتیب خطی آنها توسط اشاره گرها تعیین می گردد .

عناصر لیست تنها می توانند به ترتیب از ابتدای لیست تا انتها مورد دسترسی قرار بگیرند. هر گره آدرس گره بعدی را شامل می شود که به این صورت امکان پیمایش از یک گره به گره بعدی فراهم می شود .

برای رسم لیست پیوندی گره ها به صورت مستطیل هائی پشت سرهم رسم می شوند که توسط فلش هائی بهم متصل شده اند .



مقدار ثابت NULL برای علامتگذاری انتهای لیست در اشاره گر آخرین گره ذخیره می شود .

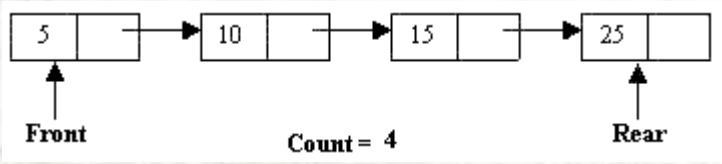
لیست توسط یک اشاره گر Head که آدرس اولین گره لیست را در خود ذخیره می کند قابل دسترس است. بقیه عناصر توسط جستجوی خطی بدست می آیند .

پیاده سازی لیست پیوندی یک طرفه

برای پیاده سازی لیست پیوندی ابتدا باید نوع داده یک گره و متغیرهای موردنیاز تعریف شوند که در زبان C می تواند به صورت زیر نوشته شود :

```
typedef int ItemType;
typedef struct Node {
  ItemType Info;
  Node * Next;
};
typedef Node * NodePtr;
NodePtr Front, Rear;
int Count;
```

در تعریف فوق ItemType نوع داده عناصر لیست را معین می کند که در مثل int درنظر گرفته شده است. ساختمان Node برای تعریف هر گره لیست است که دارای دو فیلد Info و Next است که به ترتیب عنصر داده ای گره و اشاره گر به گره بعدی را ذخیره می کنند. اشاره گر Front برای اشاره به ابتدای لیست در نظر گرفته شده است. گاهی دسترسی سریع به انتهای لیست موردنظر است، به همین دلیل ممکن است اشاره گر Rear را برای اشاره به انتهای لیست اضافه کنیم. متغیر Count تعداد گره های لیست را ذخیره می کند تا هروقت که احتیاج است بدانیم چه تعداد عنصر در لیست وجود دارد از آن استفاده کنیم .



در ابتدای برنامه متغیرهای Front و Rear باید برابر با NULL شوند تا یک لیست تهی ایجاد شود .

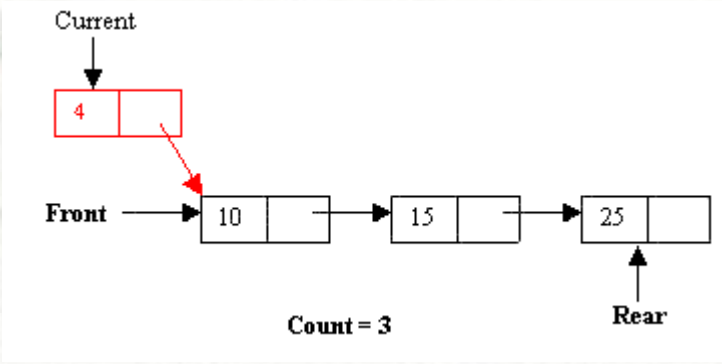
```
void CreateEmptyList(void)
{
  Front = NULL;
  Rear = Front;
  Count = 0;
}
```

برای پیاده سازی یک لیست پیوندی، عملیات اصلی شامل موارد زیر هستند:

- درج : یک گره جدید را به ابتدا، انتها یا میان لیست اضافه می شود.
- حذف : یک گره از ابتدا، انتها یا میان لیست حذف می شود.
- جستجو : یک گره که شامل مقدار خاصی است در لیست جستجو می شود.

درج یک گره در ابتدای لیست

یک گره جدید می تواند در هر جائی از لیست اضافه شود؛ ابتدا، انتها یا میان لیست. حالت زیر را درنظر بگیرید که می خواهیم گره جدید Current را به ابتدای یک لیست موجود اضافه کنیم .



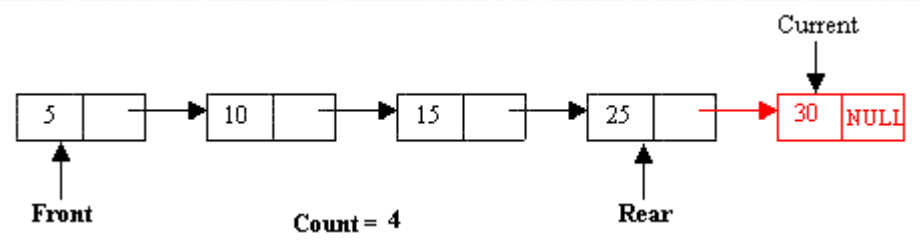
اشاره گر Current به گره جدید اشاره می کند. فیلد Next این گره باید مقدار Front مقداردهی شود تا به اولین گره لیست اشاره کند. با اضافه شدن گره جدید به ابتدای لیست اشاره گر Front باید تغییر کند و به گره ابتدا که اکنون گره Current است اشاره کند. متغیر Count در انتها یک واحد اضافه می شود .

```
void InsertFirst(ItemType item) {
  NodePtr current= new Node;
  if (current == NULL ) {
     cerr << "Memory allocation error!" << endl;
     exit(1);
     }
  current->Next = Front;
  current->Info = item;
  Front = current;
  if (Count == 0)      Rear = current;
  Count++;
}
```

در حالتی که گره جدید به ابتدای یک لیست خالی اضافه می شود(یعنی وقتی Count=0 است) باید اشاره گر Rear هم تنظیم شود تا به گره جدید که اولین و آخرین گره لیست محسوب می شود اشاره کند .

درج یک گره در انتهای لیست

برای اضافه کردن یک گره در انتهای لیست گره Current باید بعد از آخرین گره لیست که آدرس آن در اشاره گر Rear نگهداری می شود درج شود .



فیلد Next آخرین گره به گره جدید باید اشاره کند . بعد از اضافه شدن گره در انتهای لیست اشاره گر Rear برابر با آدرس گره آخر می شود .

```
void InsertLast(ItemType item) {
  NodePtr current;
  current = new Node;
  if (current == NULL) {
    cerr << "Memory allocation error!" << endl;
    exit(1);
    }
  current->Next = NULL;
  current->Info = item;
  if (Count == 0 ) Front=current;
  else             Rear->Next = current;
  Rear = current;
  Count++;
}
```

در حالتی که گره جدید به انتهای یک لیست خالی اضافه می شود(یعنی وقتی Count=0 است) باید اشاره گر Front هم تنظیم شود تا به گره جدید که اولین و آخرین گره لیست محسوب می شود اشاره کند .

وقتی لیست مرتب است گره جدید باید در محلی اضافه شود که ترتیب گره های لیست حفظ شود. بنابراین ابتدا باید محل درج گره جدید در لیست پیدا شود سپس اشاره گرها تنظیم شوند .

حذف یک گره از ابتدای لیست

عمل حذف از درج ساده تر است. البته احتمال یک پیغام خطا وقتی که لیست تهی است وجود دارد. در ساده ترین حالت حذف از ابتدای لیست صورت می گیرد .آدرس اولین گره در متغیر اشاره گر Current ذخیره می شود. مقدار فیلد Info این گره در متغیر Itemنگهداری می شود. سپس فیلد Next گره اول برای اشاره به گره دوم لیست تنظیم می شود. حافظه گره ای که از لیست حذف شده است توسط دستور free آزاد می شود. در انتها از متغیر Count یک واحد کم می شود .

یک حالت استثنا وجود دارد وقتی که لیست تنها شامل یک گره است و با حذف آن Rear باید برابر با NULL شود .

```
void RemoveFirst(ItemType Item) {
  NodePtr Current;
```

```
if (Count == 0) {
    cerr << "ERROR: there is no item to remove in the list!" << endl;
    exit(1);
  }
  Current = Front;
  Item = Current->Info;
  Front = Current->Next;
  Count--;
  if (Count == 0)     Rear = Front;
  delete Current;
}
```

جستجو در لیست

تا وقتی که لیست تهی نیست می توانیم داده معینی را در لیست جستجو کنیم .جستجو از ابتدای لیست آغاز می شود. اشاره گر Currentبرای جلو رفتن روی گره های لیست استفاده می شود و در ابتدا آدرس اولین گره در آن قرار می گیرد. در صورت وجود گره مورد نظر در لیست آدرس آن که در Current است برگردانده می شود .

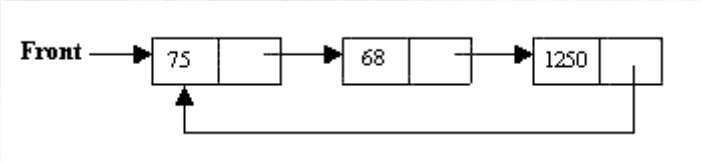
لیست پیوندی حلقوی

در مثال قبل احتیاج به یک گره آغازین و یک گره پایانی بود. ابتدای لیست توسط یک اشاره گر خاص (در مثال قبل Front و انتهای لیست توسط گره ای که اشاره گر آن برابر با NULL است مشخص می شود. اگر مسئله به عملیاتی نیاز دارد که روی یک گره لیست پیوندی انجام می شود که مهم نیست گره اول یا آخرتوسط یک لیست پیوندی حلقوی حل می شود .

لیست حلقوی (circular list)لیست پیوندی است که آخرین گره آن به اولین گره لیست اشاره می کند. لیست حلقوی اشاره گر تهی ندارد و فیلد Next آخرین گره با آدرس گره اول مقداردهی می شود .

```
Rear->Next = Front;
```

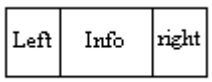
عملیات درج، حذف و جستجو مانند لیست پیوندی است.



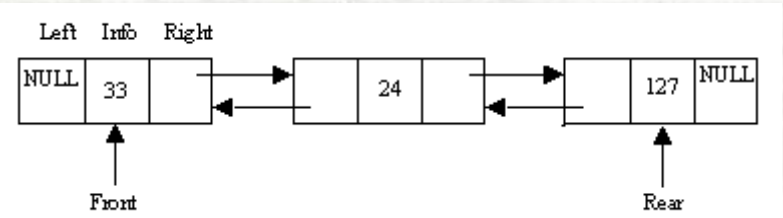
یکی از کاربردهای لیست حلقوی در اشتراک زمانی است که سیستم عامل لیستی از کاربران را دارد و به طور تناوبی به هر کاربر برش کوچکی از وقت پردازنده را اختصاص می دهد. هر بار یک کاربر را انتخاب می کند و مقداری از وقت پردازنده را می دهد سپس به کاربر بعدی منتقل می شود .

لیست پیوندی دوطرفه

در لیست یک طرفه فقط می توانیم در یک جهت پیمایش کنیم. گاهی پیمایش روی لیست از هردوطرف مورد نیاز است. بنابراین در هر گره به دو فیلد اشاره گر نیاز است؛ برای اشاره به گره بعدی و گره قبلی که اغلب اشاره گرهای Left و Right نامیده می شوند. لیستی که شامل این نوع گره باشد را لیست پیوندی دوطرفه (doubly-linked list) می نامند. شکل زیر ساختار گره را نشان می دهد .



اگر لیست به صورت افقی ترسیم شود اشاره گرهای Right برای پیمایش لیست از چپ به راست (از ابتدا به انتها) استفاده می شوند. توسط اشاره گرهای Left می توان در صورت نیاز به گره قبلی برگشت. بنابراین امکان پیمایش لیست در هردو جهت وجود دارد .



همین طور که در شکل دیده می شود اشاره گر Left اولین گره و اشاره گر Right آخرین گره لیست NULL هستند که نشان دهنده ابتدای هر جهت می باشند .

مقایسه لیست پیوندی و آرایه

مزیـ اصلی آرایه نسبت به لیست پیوندی این است که آرایه امکان دسترسی تصادفی را می دهد و می توان به هر عنصر مستقیما توسط اندیس آن مراجعه کرد .به همین دلیل در یک آرایه مرتب می توانید از جستجوی باینری به جای جستجوی خطی استفاده کنید. اما با وجودیکه آرایه امکان دسترسی سریعتر به عناصر را می دهد در عملیات درج و حذف ضعیف است و عملیات درج و حذف ممکن است باعث شیفت دادن عناصر زیاد دیگری شود. درحالیکه درج و حذف در لیست راحت تر است و درحقیقت تنها با تغییر اشاره گرها صورت می پذیرد. بنابراین احتمالا زمانی که عملیات درج و حذف زیاد انجام می شود روش بهتری است. تفاوت دیگر میزان فضای موردنیاز دو روش است. آرایه یک ساختمان داده ایستا است. هنگام تعریف، اگر اندازه آرایه را کوچک بگیریم از قدرت برنامه کاسته می شود بنابراین ناگزیریم بیشترین فضای ممکن را درنظر بگیریم که در نتیجه آرایه خیلی بزرگ تعریف می شود و حافظه زیادی مصرف می شود. درحالیکه لیست پیوندی ساختمان داده پویا است یعنی می تواند به راحتی رشد کند یا تحلیل برود یا تغییر کند. البته فضائی از حافظه برای ذخیره باید اشاره گرها صرف شود .

کلا لیست های پیوندی اغلب برای نمایش اطلاعاتی که ویژگی های زیر را دارند بکار می روند :

- تعداد کلی عناصر داده ای از قبل شناخته شده نیست.
- ممکن است عملیات اضافه و حذف زیاد انجام شوند.
- داده ها در یک طریق مرتب یا متوالی ذخیره شوند .
- با داده ذخیره شده به طور وسیع کار شود نه موردی.