به نام خداوند بخشنده مهربان

ایجاد ساختارهای داده ای در ویژوال بیسیک

گردآوری و تالیف : پیمان شاهکار (Desperado)
Email : Peyman.shahkar@Gmail.com
ID : Peyman_shahkar
Web : WWW.Desperado.coo.ir

تاریخ نگارش : ۱۳۸۷/۶/۲۳

منبع : فروم امنیتی ویرانگر

www.virangar.net

تقدیم به آنکه زندگی را به من آموخت

ایجاد ساختارهای داده ای در ویژوال بیسیک - بخش اول

مقدمه :

ساختارهای داده ای از نظر تعداد اعضا به دو دسته استاتیک و دینامیک تقسیم می شوند .
ساختارهای استاتیک مثل آرایه های یک بعدی و آرایه های دو بعدی ، تعداد اعضای آنها در زمان طراحی برنامه مشخص می شود و در طول اجرای برنامه ثابت است اما تعداد اعضای ساختارهای داده ای دینامیک در طول اجرای برنامه تغییر می کند . لیست پیوندی (LinkList) ، پشته (Stack) ، صف (Queue) و درختهای باینری (Tree Binary) ، نمونه هایی از ساختارهای داده ای دینامیک هستند .
لیست پیوندی شامل مجموعه ای از عناصر داده ای است که اضافه و حذف اعضا در هر جای لیست ممکن است .
پشته یک ساختار داده ای مهم در کامپایلرها و سیستم های عامل است که عمل اضافه و حذف عناصر از ابتدای آن انجام می شود .
صف یک ساختار داده ای است که عمل اضافه کردن از انتها و عمل حذف کردن از ابتدای آن انجام می شود .
درختهای دودویی برای جستجوی بسیار سریع ، ذخیره سازی داده ها و کامپایل عبارات استفاده می شوند .

نوع داده Variant :

نوع داده variant برای متغیرهایی بکار می رود که بطور صریح نوع آنها تعریف نشده است

مثال :

```
Dim value As Variant
```

این نوع داده می تواند هر نوع داده ای را در خود ذخیره کند . همچنین برای ایجاد ساختارهای داده ای مثل لیست های پیوندی ، صف ، پشته و درخت مناسب است .

نوع داده موجود در variant می توان توسط توابع VarType و TypeName تعیین کرد .

تابع VarType یک مقدار صحیح برمی گرداند که نشان دهنده نوع ذخیره شده در variant است .

مثال :

```
Dim value asVariant
value="Hello"x
```

در اینصورت مقدار بازگشتی VarType(value) برابر ۴ خواهد بود .

تابع TypeName یک رشته برمی گرداند که نشان دهنده نام نوع داده ذخیره شده در variant است .

اخذ حافظه بطور دینامیک Dynamic MemoryAllocation :

برای ایجاد و نگهداری ساختارهای داده ای دینامیک بایستی در هنگام اجرای برنامه بتوان فضای بیشتری برای نگهداری داده های جدید بدست آورد . با استفاده از کلمه کلیدی New می توان در ویژوال بیسیک حاقظه دینامیک گرفت :

```
SetNewNode=New ListNode
```

که ListNode یک شی از ساختار داده ای مورد نظر ماست .

کلاسهای خود ارجاعی :

کلاس خودارجاعی نوعی کلاس است که دارای یک اشاره گر (Pointer) به یک شی از همان نوع کلاس باشد . برای مثال اگر کلاس ما به اسم ClistNode باشد و متغیر زیر را در آن تعریف کنیم ، این کلاس یک کلاس خود ارجاعی است :

Private mNextNode as ClistNode

از mNextNode برای لینک دادن اعضای یک ساختار داده ای دینامیک بهم استفاده می شود (بعبارت دیگر گره زدن یک شی از کلاس ClistNode به یک شی دیگر از همان کلاس) . شی های خودارجاعی می توانند به همدیگر لینک شوند و ساختارهای داده ای مثل لیست پیوندی ، صف ، پشته و درخت را ایجاد کنند .

شکل زیر دو شی خود ارجاعی را نشان می دهد که بصورت یک لیست بهم لینک شده اند . عبارت NULL بدین معنا است که شی خودارجاعی به شی دیگری اشاره نمی کند (Nothing) و نشان دهنده انتهای ساختار داده است .

ایجاد ساختارهای داده ای در ویژوال بیسیک - بخش دوم

لیست پیوندی

همانطور که گفته شد لیست پیوندی مجموعه ای از یکسری داده است که این داده ها از نوع اشیا خودارجاعی هستند . (هر شی خودارجاعی دارای یک متغیر نوع variant برای نگهدار مقدار و یک اشاره گر به شی بعدی است) . هر عضو لیست پیوندی را یک گره گویند . هر لیست پیوندی از طریق یک اشاره گر به اولین گره قابل دسترسی است . گره های بعدی از

طریق قسمت لینک موجود در هر گره قابل دسترس هستند . همچنین لینک آخرین گره با Nothing تنظیم می شود که انتهای لیست را نشان می دهد .

مزیت اصلی لیست های پیوندی نسبت به آرایه اینست که تعداد عناصر لیست پیوندی قابل تغییر است . بعبارت دیگر لیست های پیوندی بصورت دینامیک هستند و طول آنها قابل تغییر است اما سایز آرایه ثابت است . (البته ویژوال بیسطک از آرایه های با سایز متغیر نیز پشتیبانی می کند اما این عمل تغییر سایز اتوماتیک نیست .)

عمل درج در لیست پیوندی ساده است و تنها بایستی دو اشاره گر تغییر یابد .

لیست های پیوندی را می توان به سادگی با قراردادن هر عضو جدید در محل صحیح بصورت sortشده نگهداری کرد .

اعضای لیست پیوندی در حافظه بصورت پیوسته ذخیره نمی شوند بنابراین نمی توان فوراً به هر عضو لیست دسترسی داشت (بر خلاف آرایه) .

برای ایجاد لیست پیوندی در ویژوال بیسیک نیاز به سه کلاس است :

۱ – کلاس ClistNode : کلاسی است که هر گره از لیست را توصیف می کند :

```
private mNodeData as Variant
privatemNextNode as ClistNode
public Property Get Data() asVariant
Data=mNodeData
End Property
Public Property Let Dta(ByValvNewValue as Variant)x
MNodeData=vNewValue
End Property
Public PropertyGet NextNode() as ClistNode
Set NextNode=mNextNode
End Property
PublicProperty Let NextNode(Byval vNewValue as Variant)x
SetmNextNode=vNewValue
End Property
```

۲ – کلاس Clist برای توصیف لیست پیوندی .

mFirstNode برای اشاره به اولین ClistNode و mLastNode برای اشاره به آخرین ClistNode در یک شی clist بکار می رود . زمانیکه یک Clsit ایجاد می شود این دو متغیر با Nothing تنظیم می شوند . روال Property GetIterator یک شی ClistIterator برمی گرداند که می توان از آن برای حرکت در بین اعضای لیست استفاده کرد .

```
Private mFirstNode asClistNode
Private mLastNode as ClistNode
Public Function IsEmpty() asboolean
IsEmpty=IIf(mFirstNode Is Nothing,True,False)x
Endfunction
Public Sub InsertAtFront(insertItem as variant)x
Dim tempNode asClistNode
If IsEmpty() then
Set mFirstNode=New ClistNode
SetmLastNode=mFirstNode
Else
Set tempNode=mFirstNode
Set mFirstNode=NewClistNode
MFirstNode.NextNode=tempNode
Endif
MFirstNode.Data=insertItem
End sub
Public subInsertAtBack(insertItem as Variant)x
Dim tempNode as ClistNode
IfIsEmpty() then
Set mLastNode=New ClistNode
SetmFirstNode=mLastNode
Else
Set tempNode=mLastNode
Set mLastNode=NewClistNode
TempNode.NextNode=mLastNode
Endif
MLastNode.Data=insertItem
End sub
Public functionRemoveFromFront()x
Dim removeItem as Variant
If IsEmpty() then
```

```
Msgboxlist is empty
RemoveFromFront=Null
Exit function
Endif
RemoveItem=mFirstNode.Data
If mFirstNode Is mLastNode then
SetmFirstNode=Nothing
Set mLastNode=Nothing
Else
SetmFirstNode=mFirstNode.NextNode
End if
RemoveFromFront=removeItem
Endfunction
Public Function RemoveFromBack()x
Dim removeItem asVariant
Dim current as ClistNode
If IsEmpty() then
Msgboc list isempty
RemovefromBack=Null
Exit function
Endif
RemoveItem=mLastNode.Data
If mFirstNode Is mLastNode then
SetmFirstNode=nothing
Set mLastNode=Nothing
Else
Setcurrent=mFirstNode
While Not current.NextNode Is mLastNode
Setcurrent=current.NextNode
Wend
SetmLastNode=current
Current.NextNode=nothing
Endif
RemoveFromBack=removeItem
End function
Public property GetIterator() as variant
Dim iter as ClistIterator
Set iter=NewClistIterator
Iter.StartNode=mFirstNode
Set Iterator=iter
```

Endproperty

عملکرد روال InsertAtFront :

a – فراخوانی IsEmpty برای تعیین خالی بودن لیست

b – اگر لیست خالی باشد mFirstNode و mLastNode به New ClsitNode اشاره می کنند .

c – اگر لیست خالی نباشد گره جدید توسط اشاره دادن tempNode به اولین گره لیست و سپس اشاره دادن mFirstNode به گره New ClsitNode و سپس اشاره دادن mFirstNode.NextNode به tempNode ساخته می شود .

d – تنظیم mFirstNode.Data با مقدار مورد نظر

عملکرد روال InsertAtBack :

a – فراخوانی IsEmpty برای تعیین خالی بودن لیست

b – اگر لیست خالی باشد mFirstNode و mLastNode به New ClsitNode اشاره می کنند .

c – اگر لیست خالی نباشد گره جدید توسط اشاره دادن tempNode به آخرین گره لیست و سپس اشاره دادن mLastNode به گره New ClsitNode و سپس اشاره دادن tempNode.NextNode به mLastNode ساخته می شود .

d – تنظیم mLastNode.Data با مقدار مورد نظر

عملکرد روال RemoveFromFront :

a – اگر لیست خالی باشد Null برگشت داده می شود .

b – اگر لیست خالی نباشد داده mFirstNode به removeItem اختصاص داده می شود .

c – اگر لیست فقط یک گره داشته باشد mFirstNode و mLastNode با Nothing مقدار دهی می شوند و گره از لیست حذف می شود .

d – اگر گره بیش از یک عضو داشته باشد mFirstNode برابر mFirstNode.NextNode می شود .

e – مقدار removeItem برگشت داده می شود .

عملکرد روال RemoveFromBack :

a – اگر لیست خالی باشد Null برگشت داده می شود .

b – اگر لیست خالی نباشد داده mLastNode به removeItem اختصاص داه می شود .

c – اگر لیست یک گره داشته باشد mFirstNode و mLastNode با Nothing مقدار دهی می شوند و گره از لیست حذف می شود .

d – اگر لیست بیش از یک گره داشته باشد متغیر current برابر mFirstNode می شود . سپس با استفاده از current روی گره های لیست حرکت می کنیم تا به گره ای برسیم که به آخرین گره اشاره می کند . سپس mLastNode را به گره ای که current به آن اشاره می کند قرار می دهیم و مقدار current.NextNode را Nothing می کنیم تا بعنوان آخرین گزه لیست معرفی شود .

e – مقدار removeItem برگشت داده می شود .

۳ – کلاس ClistIterator : این کلاس برای حرکت روی گره های لیست و دستکاری هر گره بکار می رود . از حرکت کننده ها برای چاپ لیست و یا انجام دادن عملی بر روی هر عضو Clist می توان استفاده کرد . این کلاس دارای دو متغیر از نوع ClistNode به نامهای mBookmark و mFirstNode است . متغیر mFirstNode به اولین گره در Clist اشاره می کند و متغیر mBookmark موقعیت فعلی حرکت کننده بر روی Clist را نشان می دهد . روال Property Let StartNode این دو متغیر را مقدار دهی اولیه می کند . تابع NextItem اگر مقدار mBookmark برابر Null باشد ، Null برگشت می دهد و در غیراینصورت مقدار tempData را برابر mBookmark.Data و مقدار mBookmark را برابر mBookmark.NextNode قرار می دهد . تابع HasMoreItems اگر لیست دارای چندین عضو باشد True برمی گرداند . روال ResetBookmark حرکت کننده را به ابتدای لیست منتقل می کند .

```
Private mBookmark asClistNode
Private mFirstNode as ClistNode
```

```
Public Property LetStartNode(Byval vNewValue as variant)x
Set mFirstNode=vNewValue
SetmBookmark=mFirstNode
End property
Public function NextItem()x
DimtempData as varaint
IfmBookmark Is nothingthen
NextItem=Null
Else
TempData=mBookmark.Data
SetmBookmark=mBookmark.NextNode
NextItem=tempData
End if
Endfunction
Public function HasMoreItems() as boolean
HasMoreItems=IIf(NotmBookmark Is nothing,True,False)x
End function
Public subResetmBookmark()x
MBookmark=mFirstNode
End sub
```

ایجاد ساختارهای داده ای در ویژوال بیسیک - بخش سوم

کلاس پشته :

همانطور که در بخش قبل گفته شد پشته نوعی لیست پیوندی است که گره های جدید فقط به انتهای آن اضافه شوند . روالهای اصلی پشته Push و Pop هستند .

Push یک گره جدید به بالای پشته اضافه می کند و Pop از بالای پشته گره ای را حذف کرده و مقدار داده آن را بر می گرداند .

یک کلاس پشته را با استفاده از کلاس Clist و بصورت زیر پیاده سازی می کنیم :

```
Private list As New Clist
Public SubPush(value as Variant)x
List.InsertAtFront(value)x
End sub
PublicFunction Pop As Variant
Pop=list.RemoveFromFront()x
End Function
PublicFunction IsStackEmpty() As Boolean
IsStackEmpty=list.IsEmpty()x
Endfunction
Public Property Get Iterator() as variant
SetIterator=list.Iterator
End Property
```

در این کلاس ابتدا یک شی از نوع کلاس Clist تعریف شده است . سپس متدهای Push توسط متد InsertAtFront و Pop توسط متد RemoveFromFront پیاده سازی شده اند .

یک برنامه نمونه :

برای نوشتن یک برنامه برای کار با پشته ابتدا کلاس Stack را که کد آن را در بالا دیدید به پروژه تان اضافه کنید . سپس در بخش کد مربوط به فرمتان ابتدا یک شی از نوع کلاس Stack بصورت زیر تعریف کنید :

```
Dim mStack as NewStack
```

سپس در فرمتان سه CommandButton با نامهای Push و Pop و ShowStack و نیز یک TextBox با نام StackMember قرار دهید .

کد زیر را برای کلیک شدن دکمه Push بنویسید :

```
mStack.push(StackMember.text)x
```

کد زیر را برای کلیک شدن دکمه Pop بنویسید :

```
StackMember.text=mStack.Pop()x
```

کد زیر را برای کلیک شدن دکمه ShowStack بنویسید :

```
Dim elements as NewClistIterator
Set elements=mStack.Iterator
If elements.HasMoreItems=falsethen msgbox "stack is empty"x
Else
While elemets.HasMoreItems
Msgboxelements.NextItem
Wend
```