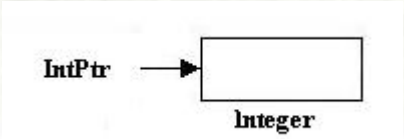
حافظه پویا و اشاره گرها

برای پیاده سازی ساختارهای پویا از اشاره گرها استفاده می شود.

اعلان متغیر اشاره گر
مقداردهی اولیه اشاره گر
عملیات اشاره گرها
تخصیص حافظه پویا
اشاره گر به آرایه
اشاره گر به رکورد

یک اشاره گر (Pointer) نوع خاصی از متغیر است که حاوی آدرسی از حافظه می باشد . درحالیکه یک متغیر استاندارد بایت هائی از حافظه را تعیین می کند که برای ذخیره نوع خاصی از داده کنار گذاشته شده است، متغیر اشاره گر به بخش هایی از حافظه که توسط متغیر دیگری اشغال شده اشاره می کند .

اشاره گرها امکان استفاده از حافظه آزاد را فراهم می کنند. علاوه براین، به طور پویا در طی اجرای برنامه می توان آنها را ایجاد یا حذف کرد .



از اشاره گرها برای ایجاد ساختمان داده های دیگر نظیر لیست های پیوندی، پشته، صف و درخت های دودوئی استفاده می شود .

اعلان متغیر اشاره گر

هر متغیر اشاره گر به نوع خاصی از داده اشاره می کند. در برنامه باید به کامپایلر اعلان شود که نوع داده ای که اشاره گر به آن اشاره می کند چیست .

در زبان پاسکال، علامت (^) قبل از نوع داده قرار می گیرد تا متغیری را به عنوان اشاره گر معرفی کند .

در زبانC ، علامت (*) برای نشان دادن اشاره گر بودن متغیری اضافه می شود. علامت ستاره را می توان بلافاصله بعد از نوع داده یا قبل از نام متغیر قرار داد .

مثل .(Pascal)متغیر اشاره گر Pxبه داده صحیحی اشاره می کند.

```
Px : ^Integer;
```

مثل .(C)اشاره گر k به یک داده صحیح اشاره می کند.

```
int *k; یا int* k;
```

علامت ستاره به معنی داده ای است که k به آن اشاره می کند .

مقداردهی اولیه اشاره گر

نکته مهم هنگام کار با اشاره گرها مقدار دهی اولیه آنهاست. اگر یک اشاره گر را فقط اعلان کنید و بدون مقداردهی از آن استفاده کنید به محل نامعینی از حافظه اشاره کند و این می تواند مشکلاتی را روی سیستم تولید کند .

یک روش کلی مقداردهی اشاره گر با مقدارصفر یا تهی (ثابت NULL در زبان C و ثابت nil در زبان پاسکال) است .

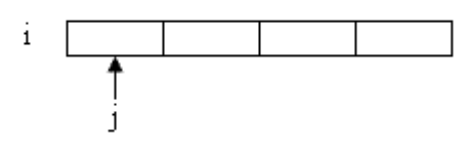
int *x=NULL; یا int *x=0;

راه معمول دیگر نسبت دادن آدرس یک متغیر استاندارد، توسط عملگر آدرس، به یک متغیر اشاره گر است.

مثل .(C)به برنامه زیر دقت کنید.

```
#include <iostream.h>
void main()
{
int i;
int* j;
j = &i;
i = 10;
cout << "i is " i;
cout << "\n j is " << j << "\n";
}
```

آدرس متغیر i توسط عملگر & بدست آمده و به اشاره گر j نسبت داده می شود، بنابراین j به متغیر i اشاره می کند i .یک متغیر صحیح است و ٤ بایت حافظه را اشغال می کند، jبه اولین بایت از این ٤ بایت اشاره می کند .



وقتی مقدار متغیر i چاپ می شود عدد ۱۰ نمایش داده می شود. با چاپ متغیر اشاره گر j یک عدد طولانی تر نشان داده می شود که آدرسی در حافظه است .

نکته. نوع اشاره گر و متغیری که به آن اشاره می کند باید یکسان باشد.

عملیات اشاره گرها

عملیات جمع و تفریق را می توان روی متغیرهای اشاره گر انجام داد. ضرب و تقسیم را روی یک اشاره گر نمی توان استفاده کرد. نکته مهمی که باید به آن توجه کرد این است که چون اشاره گر آدرسی در حافظه است وقتی عملیاتی که روی آن انجام می گیرد رفتار متفاوتی دارد. برای مثال عمل جمع اشاره گر را به تعداد بایت های نوع داده آن حرکت می دهد .

مثل .(C)چون a اشاره گری به یک عدد صحیح است و نوع صحیح ٤ بایت دارد با عمل افزایش ٤ واحد به a اضافه می شود. یعنی به ٤ بایت بعدی حافظه اشاره می کند و دیگر به همان ٤ بایت قبلی اشاره نمی کند .

```
int *a;
a++;
```

مثل .(C)اشاره گر p هشت بایت به جلو حرکت می کند و دیگر به متغیر a اشاره نمی کند.

```
int a;
int *p;
p=&a;
p=p+2;
```

تخصیص حافظه پویا

اشاره گرها زمانی نقش مهمی را در برنامه بازی می کنند که بخواهیم یک تکه از حافظه پویا (Heap) را در حین اجرای برنامه به داده ای اختصاص دهیم .ناحیه Heap فضای آزاد حافظه در دسترس است که به صورت پویا استفاده می شود، یعنی در حین اجرای برنامه در صورت نیاز اختصاص داده می شود و هنگامی که دیگر به آن احتیاج نباشد آزاد می شود .

در زبان پاسکال توابع new و dispose برای تخصیص و بازپس گیری حافظه هنگام کار با حافظه پویا استفاده می شوند .

مثل.(Pascal)

```
Var ptr:^Integer;
Begin
New(Ptr); {allocate memory to an Integer data}
{ Use ptr }
... Dispose(Ptr); {deallocate memory from the data}
End.
```

در زبان C++ دو عملگر new و delete برای اختصاص حافظه پویا بکار می روند .

دستور new تعداد بایت های معینی از حافظه پویا را به داده اختصاص می دهد. مقدار فضای مورد نیاز با توجه به نوع داده اشاره گر واگذار می شود .

اگر دستور new موفق باشد اشاره گری به فضای اختصاص داده شده بر می گرداند و اگر ناموفق باشد مقدار NULL را برمی گرداند. بعد از آن می توان از متغیر اشاره گر برای دسترسی به داده درصورت نیاز استفاده کرد .

وقتی فضای حافظه پویا دیگر مورد نیاز نباشد باید به حافظه آزاد برگردانده شود. دستور delete داده را از بین می برد و باعث می شود فضای حافظه آزاد شود تا برای مورد دیگری مجددا مورد استفاده قرار گیرد .

مثل .(C)برای اختصاص فضا به یک داده صحیح دستورات زیر نوشته می شود:

```
int *IntPtr;
IntPtr=new int;
if (IntPtr!= NULL)
* IntPtr =55;
```

اگرچه بطور ایده ال فضای پویای اختصاص یافته به برنامه بعد از اتمام برنامه باید آزاد شود اما همیشه این اتفاق نمی افتد. یک برنامه که کلیه فضائی را که گرفته آزاد نمی کند دچار memory leaks است. نتیجه آن این خواهد بود که بعد از اجرای برنامه حافظه آزاد کمتری نسبت به قبل از اجرا خواهید داشت تا زمانی که کامپیوتر را راه اندازی مجددا کنید. راه حل اصلی، پس گرفتن حافظه ای تخصیص داده شده در انتهای برنامه است .

اشاره گر به آرایه

معمولا یک اشاره گر به آرایه در برنامه زیاد مورد استفاده قرار می گیرد. اشاره گر به آرایه به اولین بایت آن اشاره می کند. هر عملگر محاسباتی که روی آن انجام شود اشاره گر را به جلو و عقب آرایه حرکت می دهد .

```
#include <iostream.h>
void main ()
{
int i[5];
int *p;
p = i;
for ( int j = 0; j<5 ; j++,p++ ) {
  *p = j;
   cout << i[j];
  cout << "\n";
}
}
```

اشاره گر به رکورد

اشاره گر به رکورد یا ساختمان اجازه دسترسی به آن را مانند هر متغیر دیگری می دهد. تنها تفاوت در اینستکه چون رکورد یک نوع داده ترکیبی است هنگام دسترسی فیلد موردنظر در آن باید تعیین شود .

مثل .(C)ساختمان account به صورت زیر تعریف شده است:

```
typdef struct account {
  float balance;
};
account *ptracout;
```

برای مقداردهی فیلد balance به صورت زیر باید عمل کرد:

```
ptraccount->balance=2000;
```

مثل .Ptr (Pascal)اشاره گری به نوع رکورد CustRec است .

```
Type
Ptr=^CustRec;
CustRec = Record
  Code:Integer;
  Name:String[25];
  Address : String[50];
End;
```

برای دسترسی به فیلد Name دستور زیر نوشته می شود:

```
Ptr^.Name := "Sara";
```