موضوع مقاله:اشاره گرها

تاریخ:۱۳۸۹/۰۶/۰٤

ارایه دهنده:آزاده منطقی

نام استاد:مهندس محمد سلیمی

www.salimiteach.com

اشاره گرها

مفهوم اشاره‌گرها يكي از اساسي‌ترين مفاهيم در دنياي برنامه‌نويسي است كه به‌مرور زمان با به‌وجود آمدن زبان‌هاي مدرن برنامه‌نويسي مفهوم كاربردي‌اش را از دست داد يا كاربرد آن كم‌رنگتر از قديم شد. اما در هر صورت دانستن آن براي يک برنامه‌نويس الزامي است.

در اين مقاله سعي شده كه اين مفهوم جذاب بسادگي بيان و شرح داده شود.

براي اين‌كه با مفهوم اشاره‌گرها آشنا شويد ابتدا مقدمه‌اي از تعريف متغيرها و نحوه حافظه اشغالي توسط آنها را بيان مي‌كنيم.

در برنامه‌هاي ما دو نوع داده وجود دارد: نوع اول داده‌هاي پايه هر زبان هستند (كه معمولا انواع مختلفي از اعداد و كاراكترها هستند)، چيزي كه نوع متغير را مشخص مي‌كند حافظه‌اي است كه متغير از حافظه اشغال مي‌كند. مثلا وقتي ما در زبان C يک داده از نوع int تعريف مي‌كنيم، ۲ بايت در سيستم‌هاي ۱۶ بيت و ۴ بايت در سيستم‌هاي ۳۲بيت حافظه اشغال مي‌كند. بگذاريد اين مفهوم را بيشتر باز كنيم.

حافظه را يكسري خانه مرتب پشت سرهم در نظر مي‌گيريم كه هر كدام نشان دهنده يك بايت هستند. وقتي مي‌گوييم متغير ما ۲ بايت حافظه اشغال مي‌كند يعني دو خانه پشت سر هم از حافظه اشغال مي‌كند. هر خانه از حافظه داراي يک عدد صحيح است كه نشان دهنده آدرس آن است و معمولا آن‌را با يک عدد در مبناي ۱۶ نمايش مي‌دهند. آدرس از ابتداي حافظه محاسبه مي‌شود مثلا خانه صفرم حافظه آدرسش برابر

0x00000000است و همين‌طور خانه ۱ برابر 0x00000001 و اين عدد به‌صورت صعودي رشد مي‌كند. مقداري از اين حافظه در اختيار سيستم‌عامل است و بقيه توسط برنامه‌هايي كه اجرا مي‌شوند اختيار مي‌شود .

هر متغيري كه تعريف مي‌كنيم در يكسري خانه حافظه قرار مي‌گيرد پس هر متغير يک آدرس دارد و هميشه بايت اول آدرس متغير است. مثلا اگر نوشتيم؛ int x يعني يک متغير از نوع x داريم كه مثلا آدرس آن

0x2243FFE1است و آدرس خانه بعدي يعني بايت دوم آن برابر 0x2243FFE2 است. وقتي شما يک مقدار را به يک متغير نسبت مي‌دهيد كامپايلر آدرس متغير را پيدا كرده و مقدار مورد نظر را بايت به بايت در حافظه مورد نظر قرار مي‌دهد .

اشاره‌گر چیست؟

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.

کاربرد اشاره‌گرها

ـ يكي از پركاربردترين موارد استفاده از اشاره‌گرها آرايه‌ها هستند. مي‌توان گفت اساس كار آرايه‌ها اشاره‌گرها هستند.

ـ كاربرد ديگر اشاره‌گرها فراخواني تابع به‌صورت ارجاع است. فرض كنيد يک تابع داريد و يكسري متغير به آن مي‌دهيد و مي‌خواهيد درون تابع روي آن متغيرها تغييراتي ايجاد شود و اين تغييرات بيرون تابع نيز ديده شوند. مثلا اگر متغير x در تابع يكي زياد شد مقدار آن در فراخواني بعدي يا استفاده‌هاي ديگر از x برابرx+۱

شود. براي اين كار يک اشاره‌گر از x را به آن تابع پاس مي‌دهيد.

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 قرارداد .

اشاره‌گرها دقيقا مثل متغيرهاي معمولي تعريف مي‌شوند با اين تفاوت كه قبل از اسم آنها يک * قرار مي‌گيرد. البته بعضي از كامپايلرها علامت * را بعد از نوع مي‌گذارند كه در باطن با روش قبلي تفاوتي ندارد مثلا

int* x وint *x هيچ فرقي با هم نمي‌كنند .

قواعد نامگذاري اشاره‌گرها مثل متغيرهاي معمولي است و مانند آنها انواعي دارند مثلا اشاره‌گري از نوع int داريم.

تفاوت اشاره‌گر نوع int **با نوع** float

پيش از اشاره به اين تفاوت بايد بگوييم كه اشاره‌گر دقيقا به چه دردي مي‌خورد و چرا از آن استفاده مي‌شود.

اشاره‌گر برخلاف متغير معمولي يک آدرس را در خود ذخيره مي‌كند و نوع اشاره‌گر مشخص مي‌كند آدرس مورد نظر چند بايت است، مثلا

int *ptrXآدرس يک شروع ? خانه از حافظه را در خود ذخيره مي‌كند. براي سادگي بيان مي‌گوييم كه ptrX به يک int اشاره مي‌كند. وقتي ptrX تعريف مي‌شود به خانه‌اي اشاره نمي‌كند براي اين‌كه به ptrX بگوييم به كجا اشاره بكن بايد آدرس يک متغير را به آن بدهيم، آدرس را با & نشان مي‌دهند. مثلا&x يعني آدرس متغير .x

حال ما مي‌خواهيم يک مقدار اشاره‌گر تعريف كنيم كه به يک float اشاره مي‌كند و آدرس متغيري از نوع float را در آن قرار دهيم:

```
Float x,*ptrx;

Ptrx = &x;
```

بسيار خب از اينجا به بعد هر جا كه مقدار ptrx تغيير كرد مقدارx هم تغيير مي‌كند، چون هر دو يک فضا از حافظه را نشان مي‌دهند. در واقع ما يک كپي از متغير x داريم كه هر وقت مقدار آن‌را تغيير داديم مقدار اصلي x نيز تغيير مي‌يابد.

براي اين‌كه يک مقدار عددي را به اشاره‌گر بدهيم و در واقع به آن بگوييم كه اين مقدار را در آن خانه از حافظه كه به آن اشاره مي‌كني قرار بده از عملوند * استفاده مي‌كنيم. مي‌خواهيم مقدار ۱۰ را به متغير x نسبت دهيم و اين كار را هم از طريق ptrx انجام دهيم. اين كار به‌صورت زير انجام مي‌شود.

```
*ptrx = 10;
```

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

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

int *x=NULL; یاint *x=0;

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

مثال(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 .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 .

مثال .(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++;
```

مثال .(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 .

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 .

مثال (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 .

مثال (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;
```