مبانی کامپیوتر و برنامه سازی

استاد: ساسان حریفی

اشاره گرها

- از مهمترین اجزاء زبان c هستند. به طور کلی در سه زمینه زیر کاربرد دارند:
  - در فراخوانی توابع
  - در استفاده پویا از حافظه
  - افزایش کارایی برخی برنامه ها

- یک اشاره گر حاوی آدرس محلی از حافظه است که ممکن است در آن محل متغیر دیگری وجود داشته باشد. در این حالت می گوییم، متغیر اشاره گر به آن متغیر اشاره می کند.

اشاره گرها

- مثال)

آدرس	محتوای حافظه
1000	1004
1001	2000
1002	…
1003	…
1004	…

- در مثال فوق در آدرس 1000 متغیری وجود دارد که آدرس محل دیگری از حافظه را در خود نگه داشته است.

اشاره گرها

- نحوه تعریف اشاره گر:

نوع اشاره گر علامت ستاره نام اشاره گر ;

- مثال)

```
int   * p ;
char  * s ;
```

اشاره گرها

- مثال)

نکته) همانطور که بیان شد، اشاره گر حاوی آدرس متغیرها می باشد. چگونه آدرس را در یک متغیر اشاره گر قرار می دهیم؟ از طریق &

```
int   *p , x ;
x = 10 ;
p = & x ;
cout << *p ;
```

- در مثال فوق، خروجی عدد ۱۰ است. زیرا دستور p=&x آدرس x رادر p قرار می دهد و از این به بعد p به x اشاره می کند. در نتیجه *p مقدار x را نشان می دهد.

اشاره گرها

- نکته اول) نوع داده ای اشاره گر و نوع داده ای متغیری که قرار است آدرس آن در اشاره گر قرار گیرد باید یکی باشند.

- نکته دوم) اگر دو اشاره گر از یک نوع باشند می توان مقدار یکی را با دستور انتساب در دیگری قرار دارد.

```
int *p,*q,x;
p = &x;
q = p;
```

اشاره گرها

- نکته سوم) تنهای عملیات ریاضی قابل انجام بر روی اشاره گرها عملیات جمع و تفریق است. میزان واقعی جمع و تفریق به نوع داده ای اشاره گر بستگی دارد. یعنی اگر فرض کنید p به آدرس 2000 اشاره می کند و از نوع int باشد، با دستور p++ مقدار p به عدد 2002 افزایش می یابد چون p دو بایتی است.

- نکته چهارم) مقایسه دو اشاره گر امکان پذیر است. مثلا کوچکتر بودن اشاره گر p از q یعنی p به محلی پایین تر از حافظه اشاره می کند.

اشاره به اشاره گر

- در حالت معمولی، اشاره گر حاوی آدرس متغیری بود که مقداری را نگهداری می کرد. در حالت اشاره به اشاره گر، یک اشاره گر تعریف می شود که حاوی آدرس اشاره گری است که آن اشاره گر به متغیر معمولی اشاره می کرد.
- برای تعریف اشاره گر به اشاره گر از دو علامت ** استفاده می شود.

```
int  **p ;
```

اشاره به اشاره گر

- مثال)

```
int  x , *p , **q ;
x = 10 ;
p = & x ;
q = & p ;
cout << **q ;
```

اگر اشاره گر خالی تعریف می‌کنید بهتر است مقدار آن را برابر با null قرار دهید.

تابع (برنامه فرعی)

- برای اینکه برنامه های بزرگ قابلیت مدیریت و اشکال زدایی راحت تر داشته باشند، معمولا آن را به بخش های کوچکتر تبدیل می کنند.

- به هر کدام از بخش ها تابع گفته می شود.

- تابع ممکن است در طول برنامه بارها و بارها فراخوانی شود و عملیات تکراری بر روی داده های مختلف انجام دهد.

- برنامه های فرعی که به آن تابع نیز گفته می شود، می توانند مقدار برگشتی داشته باشد و یا اینکه هیچ مقداری را بر نگردانند.

تابع (برنامه فرعی)

- برای تعریف و استفاده از تابع دو راه وجود دارد:
  - راه اول) تعریف الگوی تابع + نوشتن تابع اصلی برنامه + نوشتن بدنه تابع
  - راه دوم) نوشتن بدنه تابع + نوشتن تابع اصلی برنامه

- نحوه تعریف الگوی تابع:

نوع تابع نام تابع (نوع پارامترهای تابع در صورت وجود) ;

تابع (برنامه فرعی)

- مثال هایی از تعریف الگوی تابع:

```
int  f1 ( int ) ;
float  f2 ( int , float );
int  f3 ( int[] , int );
void f4 () ;
void f5 ( int ) ;
```

- بدنه تابع بسیار شبیه به بدنه تابع main است که تا به امروز با آن کار می کردیم.

تابع (برنامه فرعی)

- مثال تعریف بدنه تابع:

```
(پارامترهای تابع در صورت وجود)  نام تابع   نوع تابع
{
        متغیرها و دستورات درون تابع
        ...
}
```

- برای فراخوانی تابع کافیست نام آن را صدا بزنیم، و اگر پارامتر ورودی دارد برای آن تعیین کنیم.

تابع (برنامه فرعی)

- نکته) برای برگشت دادن مقادیر از تابع، از دستور return استفاده می شود. این دستور دو وظیفه اصلی دارد:

  - این دستور در هر جای تابع اجرا شود باعث قطع اجرای تابع شده و باعث بازگشت به تابع فراخوان می گردد.

  - برای برگرداندن مقدار تابع در صورتی که مقدار برگشتی داشته باشد استفاده می شود.

تابع (برنامه فرعی)

- برنامه ای بنویسید که عددی را از کاربر دریافت کند.
  - اگر عدد ۱ باشد تابعی را فراخوانی کند که درآن تابع دو عدد دریافت شوند و حاصلضرب آن نمایش داده شود.
  - اگر عدد ۲ باشد تابعی را فراخوانی کند که در آن نام کاربر را گرفته و نمایش دهد.
  - اگر عدد، عددی غیر از ۱ یا ۲ باشد تابعی را فراخوانی کند در در آن کلمه ERROR را چاپ کند.