آموزش ++C - درس ۳: ثابتها

به سری آموزشهای ++C خوش آمدید. این درس به شما یاد خواهد داد که چگونه ثابتها را در ++C اعلان کرده و از آنها استفاده کنیم. یک ثابت، در این مفهوم که مکانی از حافظه است، شبیه به یک متغیر است. البته ثابتها فرق دارند و من مطمئنم که شما می توانید تفاوت را حدس بزنید، که پس از مقداردهی اولیه نمی توان دوباره یک مقدار جدید برای آن تعریف کرد. بطور کلی ثابتها ویژگی مفیدی هستند که می توانند از اشکالات برنامه ها و خطاهای منطقی جلوگیری کنند. از تغییرهای تصادفی نیز جلوگیری می کنند. کمپایلر مقداردهی کردن دوباره به ثابتها را متوجه خواهد شد.

تعریف کردن ثابتها در ++C
برای تعریف ثابتها در ++C سه روش مورد استفاده قرار می گیرد. اولین روش از زبان C می آید. ثابتها می توانند بوسیله جهت دهنده پیش کمپایلر یعنی #define تعریف شوند. پیش کمپایلر یک برنامه است که پیش از کمپایل شدن تغییرات لازم را در فایل سورس شما انجام می دهد. عمومی ترین جهت دهنده های پیش کمپایلر عبارتند از:

#include: که برای اضافه کردن کدهای اضافی به فایل منبع شما است.
#define: که برای تعریف یک ثابت استفاده می شود.
#if/#endif: که برای تعیین شرطی استفاده می شود و قسمتی از کد شما کمپایل خواهد شد.

مثالهایی از استفاده از جهت دهنده #define را مشاهده می کنید:

```
#define pi 3.1415
#define id_no 12345
```

هر کجا که ثابت مورد نظر در فایل منبع شما استفاده شده باشد، بوسیله پیش کمپایلر با مقدار آن جایگزین می شود. بنابراین برای مثال، هر pi در فایل منبع شمل با مقدار 3.1415 جایگزین خواهد شد. کمپایلر فقط مقدار 3.1415 را در کد شما خواهد دید نه pi را. مشکل این روش این است که این جایگزینی بصورت لغوی انجام می شود، بدون هیچگونه بررسی نوع داده، بدون هیچگونه بررسی حد و مرز و بدون هیچگونه بررسی میدان دید. هر pi فقط با مقدار آن جایگزین می شود. این روش قدیمی شده است اما برای پشتیبانی از کدهای قدیمی در حال حاضر نیز پشتیبانی می شود ولی باید از آن دوری کرد.

Const
دومین روش استفاده از لغت کلیدی Const هنگامی که یک متغیر را تعریف می کنید، می باشد. وقتی که این روش استفاده شد کمپایلر مبادرت به اصلاح متغیرهایی که بصورت Const اعلان شده اند خواهد کرد.

```
const float pi = 3.1415;
const int id_no = 12345;
```

در اینجا دو فایده نسبت به روش اول وجود دارد. اول، نوع ثابت هم تعریف شده است. Pi از نوع float است. Id_no از نوع int است. این به کمپایلر اجازه می دهد تا عمل بررسی نوع داده را انجام دهد. دوم، این ثابتها با میدان دید معین تعریف شده اند. میدان دید یک متغیر (ثابت) به قسمتهای مختلف برنامه می گوید که در کدام قابل شناسایی است. بعضی از متغیرها ممکن است فقط در توابع معین یا در بلاکهای معینی از کد وجود داشته باشند. شما ممکن است بخواهید از id_no در یک تابع و یک id_no دیگر در تابع main استفاده کنید. متاسفم اگر این مسئله کمی گیج کننده است، در مورد میدان دید متغیرها در دروس آینده بحث شده است. من در اینجا مشکل مرغ و تخم مرغ را دارم و راه حل من این بود که مرغ همان ثابت است و میدان دید هم یک متغیر.

لیست برداری
روش سوم لیست برداری (Enumerations) نامیده می شود(در بعضی از مستندات فارسی این کلمه به شمارشی ترجمه شده است-مترجم). لیست برداری یک نوع داده جدید است و مقادیر این نوع به مجموعه ای از مقادیر تعریف شده توسط برنامه نویس محدود شده است. برای مثال:

```
enum COLOR { RED, BLUE, GREEN};
enum SHAPE {SQUARE, RECTANGLE, TRIANGLE, CIRCLE, ELLIPSE};
```

هر ثابت لیست برداری شده (بعضی وقتها لیست هم نامیده می شود) یک مقدار از نوع عدد صحیح دارد. مگر اینکه تعیین شده باشد. اولین ثابت مقدار 0 دارد. مقادیر به ازای هر ثابت در لیست یکی افزایش می یابد. بنابراین ، مقدار RED 0 است و BLUE مقدار ۱ و GREEN مقدار ۲ دارد. SQURE برابر 0 و RECTANGLE برابر ۱، Triangle برابر ۲ و به همین ترتیب. مقدار هر ثابت می تواند معین شده باشد.

```
enum SHAPE {SQUARE=5,RECTANGLE,TRIANGLE=17,CIRCLE,ELLIPSE};
```

در اینجا، SQURE برابر ۵، RECTANGLE برابر ۶، TRIANGLE برابر ۱۷، CIRCLE برابر ۱۸ و ECLIPSE برابر ۱۹ است. مزیت روش لیست برداری این است که اگر متغیری از نوع یک لیست تعریف شده باشد، نوع آن مشخص است، مقادیرش محدود است و در طی کمپایل شدن بررسی می شوند. سعی کنید برنامه زیر را کمپایل کنید. چه خطایی در آن است؟

```
#include <iostream>
using namespace std;

int main()
{
    enum color {RED, GREEN, BLUE};
    color myColor = 5;

    if (myColor == RED) {
        cout << "hot" << endl;
    }
    if (myColor == BLUE) {
        cout << "cold" << endl;
    }
    if (myColor == GREEN) {
        cout << "Is not easy being" << endl;
    }

   return 0;
}
```

در برنامه قبلی مقدار myColor برابر ۵ قرار داده شده است. این کار یک خطای کمپایل می باشد. دلیل آن هم این است که myColor از نوع color تعریف شده است و مقادیر نوع color محدود به آنهایی است که در تعریف آن مشخص شده است. فقط مقادیر RED و GREEN و BLUE مجاز هستند. ۵ یک مقدار مجاز برای نوع داده color نیست. اصلاح شده برنامه قبل بدین شکل است:

```
#include <iostream>
using namespace std;

int main()
{
    enum color {RED, GREEN, BLUE};
    color myColor = RED;

    if (myColor == RED) {
        cout << "hot" << endl;
    }
    if (myColor == BLUE) {
        cout << "cold" << endl;
    }
    if (myColor == GREEN) {
        cout << "Is not easy being" << endl;
    }

   return 0;
}
```

ترجمه: محمد صافدل
منبع: http://cplus.about.com