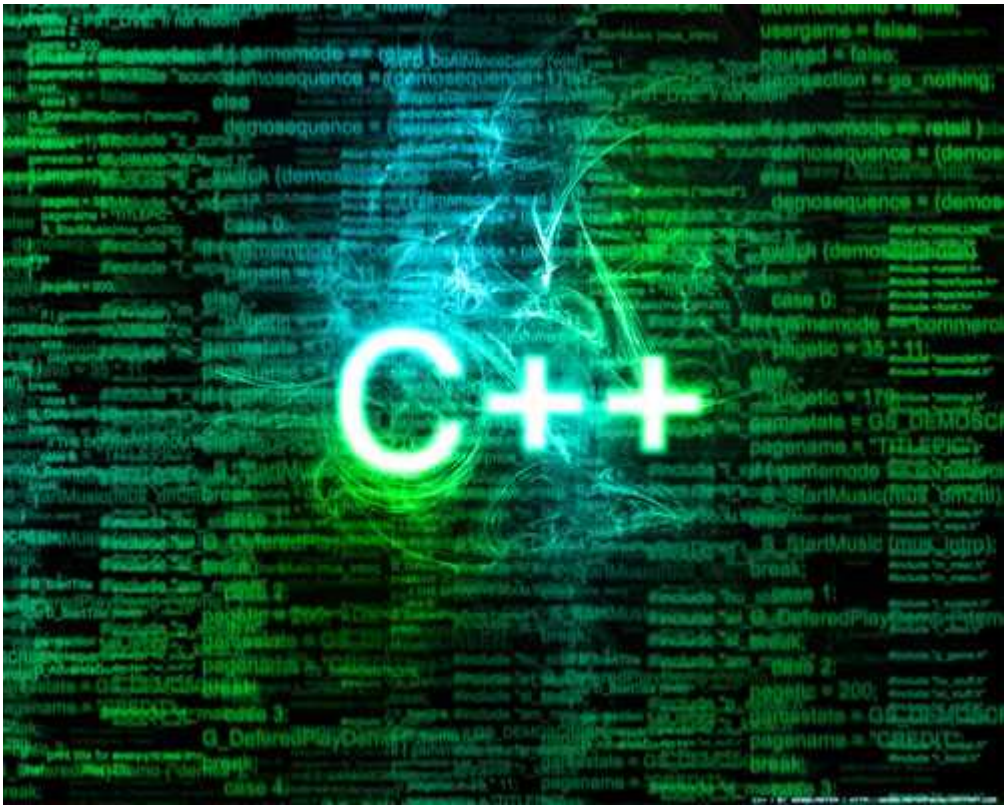
کلاس های حافظه درC++



زبان برنامه‌نویسی C++ از چهار کلاس حافظه مختلف برای تعریف متغیرها پشتیبانی می‌کند.

کلاس حافظه اتوماتیک (auto):

این کلاس اصلی‌ترین کلاس حافظه زبان C++ محسوب می‌شود. متغیرهایی که توسط این کلاس تعریف می‌شوند، با خروج از محدوده تعریف به طور خودکار از بین می‌روند. بنابراین تمامی متغیرهای عادی از این نوع کلاس هستند. یعنی شما برای مشخص کردن کلاس حافظه اتوماتیک نیاز به انجام کار خاصی ندارید. اما برای تاکید بر اتوماتیک بودن کلاس حافظه، می‌توانید از کلمه کلیدی auto استفاده کنید. به عنوان نمونه، دو عبارت زیر هم ارز هستند:

```
int n;
```

```
auto int n;
```

کلاس حافظه اتوماتیک (auto):

این کلاس اصلی‌ترین کلاس حافظه زبان C++ محسوب می‌شود. متغیرهایی که توسط این کلاس تعریف می‌شوند، با خروج از محدوده تعریف به طور خودکار از بین می‌روند. بنابراین تمامی متغیرهای عادی از این نوع کلاس هستند. یعنی شما برای مشخص کردن کلاس حافظه اتوماتیک نیاز به انجام کار خاصی ندارید. اما برای تاکید بر اتوماتیک بودن کلاس حافظه، می‌توانید از کلمه کلیدی auto استفاده کنید. به عنوان نمونه، دو عبارت زیر هم ارز هستند:

```
int n;

auto int n;
```

کلاس حافظه استاتیک (static):

متغیرهای کلاس حافظه استاتیک تا اتمام برنامه حافظه اختصاصی خودشان را حفظ می‌کنند؛ حتی اگر کنترل برنامه به خارج ناحیه تعریف آنها منتقل شود. مثال زیر را در نظر بگیرید:

```
#include< iostream.h >

void func( )

{

static int y = 1;

cout << y << "t";

y *= 2;

}
```

```
void main( )

{

func( );

func( );

func( );

}
```

خروجی برنامه:

1 2 4

در اولین فراخوانی تابع func، برای متغیر y حافظه اختصاص داده می‌شود. خط آخر تابع مقدار ۲ را در y قرار می‌دهد. اما با خروج از تابع متغیر از بین نمی‌رود. بار بعد که تابع فراخوانی شد، مقداردهی اولیه y (یعنی ۱) در نظر گرفته نمی‌شود. در نتیجه عدد ۲ چاپ می‌شود و الی آخر. حافظه اختصاصی برای y فقط زمانی آزاد خواهد شد که اجرای برنامه تمام شود.

توجه: استفاده از دستور جداگانه برای مقداردهی اولیه متغیر استاتیک خطای منطقی محسوب می‌شود:

```
#include < iostream.h >

void func( )

{

static int y;

y = 1;
```

```
cout << y << "t";

y *= 2;

}

void main( )

{

func( );

func( );

func( );

}
```

خروجی برنامه:

```
1 1 1
```

کلاس حافظه ثبات (register):

زمانی که متغیری از نوع کلاس حافظه ثبات تعریف شود، حافظه برای متغیر به جای حافظه اصلی (RAM) از حافظه پردازنده مرکزی (CPU) اختصاص داده می‌شود. در نتیجه سرعت خواندن و نوشتن متغیر بالا می‌رود. از این نوع متغیرها عموما برای شمارنده حلقه‌ها استفاده می‌شود، تا سرعت اجرای حلقه بالاتر رود.

```
void main( )

{
```

```
register int i;

for( i = 0 ; i < 10 ; i++ )

{

cout << i * i << "t";

}

}
```

توجه: استفاده از کلاس حافظه ثبات یک درخواست محسوب می‌شود. یعنی ممکن است بنا به هر دلیلی تخصیص حافظه از CPU امکان نداشته باشد. در این حالت کلاس حافظه در نظر گرفته نمی‌شود. در ضمن چون حافظه پردازنده آدرس به فرم حافظه اصلی ندارد، اشاره‌گر به این نوع متغیر قابل تعریف نیست. در نتیجه از اپراتور & هم برای دسترسی به آدرس حافظه نمی‌توان استفاده کرد.

کلاس حافظه خارجی (extern):

پروژه‌های بزرگ همیشه از چندین فایل تشکیل می‌شوند. برای دسترسی به متغیری از فایل دیگر باید از کلاس حافظه خارجی استفاده کنید. به عنوان مثال:

file1:

```
int n;
```

file2:

```
extern int n;

cout << n;
```

توجه:

۱ - کلاس حافظه خارجی برای متغیرهای عمومی استفاده می‌شود.

۲ - دو فایلی که با هم تبادل متغیر می‌کنند باید از یک پروژه باشند.