C#의 특징

- 포인터가 필요없다.

- 자동 메모리 관리

- 열거형, 구조체, 클래스 속성도 공식 문법 구조에 포함

- C++ 에서와 같이 사용자 지정 형식 연산자를 쉽게 오버로드 할 수 있다.

- 인터페이스 기반 프로그래밍 기술 오나벽 지원

- 어트리뷰트를 이용한 애스펙트(aspect)기반 프로그래밍 기법 완벽 지원

.net 플랫폼과 함께 발표된 c#언어를 이해하는데 있어서 가장 중요한 점은 아마도 c#으로는 .net런타임에서 실행되는 코드만 만들 수 있다는 것이 아닐까 쉽다.(c#으로는 com서버나 전통적인 win32 응용프로그램을 절대로 만들 수가 없다.

.net 런타임을 대상으로 해서 만들어진 코드를, 공식 용어로 관리코드라고 한다. 또 관리 코드를 포함하는 바이너리 유닛을 어셈블리라고 부른다.

**어떤 .net 언어를 선택해서 프로그래밍을 하든 기억해 두어야 할것이 하나 있는데, 그것은 .net 바이너리 파일의 확장자가 기존의 com바이너리 파일의 확장자와 동일하긴 하지만(.dll 또는 .exe), 양자 사이에 어떠한 내적 유사성도 없다는 사실이다.**

Dll을 웹서비스형식으로 호출해서 쓸수도 있다

(해당기술이 지원안되는 c++은 어떻게 구현할지 골머리군요.)

80포트를 이용해 XML로 웹서버와 통신하기도 쉽다.

어플리케이션을 웹에도 띄울수 있는 스마트 클라이언트

성능 상의 문제에 관한 미션 크리티컬 이슈를 내포하는 경우 : 수학, 금융, 과학, 그래픽, 게임, 3D 그래픽 등 고속 연산을 기초로 하는 어플리케이션들은 어셈블리, C, C++같은 언어를 이용하는 것이 적당합니다.

Long-Run Application을 작성하는 경우 : 안정성이 중시되어야하고, 메모리 순환 가능성을 고려해야 하거나, 단 기간내에 다양한 요구 사항을 만족하는 복잡한 어플리케이션을 작성해야 할 경우 C#이 좋을 것입니다. 그러나 간혹 C/C++과의 연동을 위하여 C#에서만 사용이 가능한 unsafe context를 택하여 부족한 부분도 보충할 수 있을 것입니다.

공통분모 : 이벤트 기반 프로그래밍, 객체

2009년 10월3일 배우는 언어 순서 동향

C -> API -> MFC : 잘못된 선택

C++ -> C#(MFC) -> Com or ASP.net : 올바른 선택