**프로그래밍의 20가지 법칙**

**1. 프로그램의 치명적인 실행 중단은 절대로 용납하면 안 된다.**

이는 가장 중요한 법칙이다. 사용자만이 프로그램을 중지시킬 권한이 있다. 프로그램이 중단되려고 하면, 프로그램에서는 에러를 찾아 사용자에게 알려주고, 사용자가 데이터를 저장할 수 있게 한 후에 중지 버튼을 눌러 프로그램을 종료시키도록 해야 한다. 이는 곧 운영체제가 사용자에게 직접 에러 메시지를 보내서는 절대로 안 된다는 뜻이다. 운영체제는 항상 치명적인 중단을 한다.

**2. 사용자 매뉴얼, 명세서, 도움말, 소스 코드 순서로 일하라**

이 과정을 따라 프로그램을 설계할 때쯤 되면 이미 계획서가 만들어져 있다. 사용자 매뉴얼을 쓸 때 사용자들의 참여를 요구하라. 이렇게 하면, 더욱 일관성 있게 마감 시간을 지킬 수 있을 것이고, 사용자들은 그들이 원하는 대로 설계가 된다는 사실에 기쁨을 느낄 것이다.  
원하는 곳에 기술적인 문서와 같은 다른 요소를 넣어도 되지만, 사용자 매뉴얼, 명세서, 도움말, 소스 코드 순서는 항상 같아야 한다.

**3. 위험 요소 분석(RFA)을 사용하지 않으면, 프로그램 개발은 계속해서 생각하는 것보다 2배는 더 걸린다.**

그렇다. 계속해서 2배 더 걸린다. 만일 연구를 위해 프로젝트 시간을 2배로 늘린다고 해도 문제가 생길 것이다.  
RFA가 해결하는 문제는 연구 시간이다. 모든 프로그래머들은 연구가 얼마나 걸릴지를 과소평가한다. 우리는 프로젝트 내의 모든 것을 할 수 있는 기회가 별로 없다. 그러므로 우리가 어떻게 해야 하는지 모르는 일이 얼마나 걸릴지도 예상해야 한다! RFA는 이러 연구 시간에 대한 답을 준다.

**4. 코드 짜는 부분은 개발의 20%를 넘지 않아야 한다.**

좋은 설계를 가지고 있는 경우, 실제 코드를 쓰는 부분은 전체 개발의 10%만 차지할 수도 있다! 개발 노력의 20% 이상을 코딩에 쓰고 있다면, 그 원인은 거의 잘못된 초기 설계 때문 일 것이다.

**5. 검사는 적어도 프로젝트의 30%는 차지해야 한다.**

프로젝트 시간을 예상할 때 검사를 위한 30%를 추가하라. 자동화된 검사를 하더라도 검사를 위해 30% 소비해야 한다. 자동화된 검사는 더 완벽하게 검사할 수 있도록 해주지만, 그것만으로 충분하지는 않다. 만일 마감 시간 때문에 프로그램 테스트에 30%정도의 시간도 할당해 두지 않는다면, 이는 프로그램에 버그를 만들며, 프로그램을 문서화하는데도 오류를 남기게 된다.

**6. 주석은 소스의 20%를 차지해야 한다.**

유지는 그것이 문제점을 고치는 것이든, 향상할 부분을 추가하는 것이든, 프로그램 수명의 반을 소비한다. 모든 변수 선언을 문서화해서 다음 프로그래머의 작업이 쉬워지도록 한다. 함수의 이름으로 그 함수의 기능을 충분히 알아볼 수 없는 한, 중요한 함수에는 그 목적을 써줘야 한다. 주석에는 저자, 버전 날짜, 그리고 자주 빼먹는 수정 일지가 들어있어야 한다.

주석은 대부분을 /// 형식으로 짜는 것을 고려하라. VS.NET에서 직접 입력해야 할 내용을 많이 줄여줄 것이다. .

7. 에러 메시지는 무엇이 일어났는지, 사용자가 무엇을 할 수 있는지, 프로그램이 다음에 무엇을 할 것인지, 코드의 어느 줄이 문제를 일으켰는지를 알려줘야 한다. 시간, 사용자 이름, 그리고 환경도 알려줄 수 있다.

보통, 에러 메시지로 통하는 것이 일반적으로 에러 메시지의 제목에 들어간다! 불쌍한 사용자가 프로그램 개발자를 증오하지 않고 사랑할 수 있게 하라.

**8. 좋은 프로그램들은 자동으로 최근의 에러 메시지를 영구적인 매체에 저장해 둔다.**

메시지들을 원형 큐에 넣어 메시지가 디스크 공간을 차지하지 않도록 하라. ASCII 텍스트를 사용하여 아무 편집기나 워드 프로세스에서 볼 수 있도록 한다.

사용자가 프로그램 개발자에게 메시지를 보낼 수 있는 방법을 제공한다. 버튼 클릭 하나로 할 수 있다면 더욱 좋다.

**9. 루틴을 세 번 부른다면? 숨겨라. 한 번 부른다면? 숨기지 말라.**

단 한 번 사용하는 루틴은 숨기지 말라. 소스 코드에 inline 으로 추가하라. 그러나 루틴을 여러 번 사용한다면 함수로 만들어 묘사적인 이름을 붙여라.

**10. 루틴은 단 하나의 진입점과 탈출점이 있어야 한다.**

예외에는 메뉴와 에러 잡는 코드가 있다.

Structured Programing 의 저자들 중에는 그 누구도 goto 문의 결여를 받아들인 사람이 없다. 대신, 그들은 루틴에 단 하나의 진입점을 요구했다. 그들은 메뉴나 에러 잡는 코드와 같은 예외적인 경우만 여러 개의 탈출점을 허용했다.

C#은 goto 문이 필요 없을 정도로 이 예외를 잘 처리한다.

**11. 변수와 루틴을 위한 명확한 이름으로 코드를 문서화하라.**

가장 좋은 문서화는 변수, 함수, 클래스, 그리고 패키지의 이름이다. 관습을 채택해도 좋다. 일반 프로그래머들 사이에는 헝가리언 표기법이 선호되지 않지만, 이것을 쓰면 변수형을 바르게 할 수 있다. 또, 이 방식은 프로그램 수명 기간의 반을 책임지는 유지 프로그래머들을 돕는다.

**12. 관계 데이터베이스를 쓴다.**

작은 파일들과 에러 메시지를 기록하는 파일은 순차적인 파일이어도 된다. 그러나 만일 데이터를 많이 쓰고자 한다면, 그 테이블을 위한 관계 데이터베이스를 작성할 시간의 여유를 둔다. 이렇게 하면, 일반적으로 개발 시간을 줄이고 디스크 공간을 덜 소비하며, 성능을 향상시킬 수 있을 것이다.

가장 큰 이익은 프로그램 향상 기간에, 특히 파일을 관계 데이터베이스로 변환해야 할 때 나타난다. 변환한다는 의미는 주로 프로그램 전체를 다시 짜야한다는 뜻이다.

**13. 항상 제일 좋은 알고리즘을 사용하라.**

"제일 좋은" 성능, 디스크 공간 사용량, CPU 사용량, footprint 등으로 결정된다. 그러나 "가장 나쁜"과 "가장 좋은"의 차이는 1000:1일 수 있다.

**14. 가장 느린 루틴을 먼저 최적화하라. Profiler로 이들을 찾아낸다.**

이 작업은 여러분의 에너지를 가장 잘 활용할 수 있는 곳에 집중하는 일이다. 100초 걸리는 루틴을 반으로 줄이는 일을 미루고, 1초 걸리는 루틴을 반으로 줄이는데 매달리는 것은 말이 안 된다.

**15. 가장 좋은 언어는 가장 적은 개발 시간을 필요로 하는 언어이다.**

여기에서 중요한 조건이 있다. 여러분이 기존의 프로그램을 업그레이드 하는 중이면, 그것을 작성할 때 사용했던 언어를 유지하도록 노력한다. 이를 통해 시간적 우위를 가질 수 있다. 그러나 프로젝트를 지원하느냐 안 하느냐의 문제에서 제품을 시장에 낼 때까지 걸리는 시간은 매우 큰 요소이다. 그러므로 새로운 언어로 인해 개발 시간을 최단으로 줄일 수 있다면, 그 언어를 선택해야 한다.

한 가지 플랫폼 이상에서 돌아가는 프로그램을 개발한다면, 이는 C#를 선택할 충분한 동기가 된다.

**16. 사용자 서명을 요구하라.**

계약서 없이 집을 살 수 있는가? 중고차를 그런 식으로 살 수 있겠는가? 그렇다면 해야 하는 일을 확실히 알지도 못하면서 왜 이것들보다 훨씬 비싼 컴퓨터 프로그램을 만들고자 하는가? 결국, 계약서는 이해를 명확하게 결정짓는 도구에 불과하다. 이는 소송을 위한 준비 작업이 아니다. 계약서에는 여러분과 사용자가 각각 무엇을 할 것인지 명기하고 있으므로 나중에 다시 참조해도 된다.

**17. 위험한 모듈들부터 작성하라.**

이렇게 하면, 나쁜 소식을 먼저 알아낼 수 있어서 좋고, 나중에 고치는 비용이 덜 든다.

**18. 유지하기 편하도록 하라.**

주석을 자세하게 다는 것만으로는 충분하지 않다. 간단하게 명확한 알고리즘을 사용한다. 복잡한 과정들이 나올 때는 이름을 잘 붙인 임시 변수 여러 개를 사용하여 단계별로 주석을 쓸 수 있도록 한다. 이런 변수들은 유지 프로그래머가 중간 값을 점검하고, 어느 곳에서 프로그램이 이상하게 되었나를 알아낼 때 도움이 된다.

**19. 여러분이 쓰는 모든 것의 철자를 점검하고 서명하라.**

여러분이 하는 일에 자부심을 가지고 서명하라. 어느 날 누군가가 전화해서 여러분이 했던 멋진 일에 대해 축하해 줄 것이라는 희망을 가져라. 그것은 이 세상에서 가장 좋은 느낌이다.

**20. 언제가 끝인지 알자.**

범위에 대한 두려움은 프로젝트의 독이다. 피하라. 거부하라. 다음 단계로 넘겨라. 그러면 프로젝트를 다 끝내고 한숨을 돌리며 말할 수 있다. "이 프로젝트는 끝났다!" 그리고는 커피 한 잔을 마시고 프로젝트를 넘기고 향상 점들에 대해 의논하라.

**[출처]** [프로그래밍의 20가지 법칙](http://blog.naver.com/dhdma1/3740375)|**작성자** [파코파코](http://blog.naver.com/dhdma1)