200 가지의 "안녕, 세상아!"

서울대학교 컴퓨터공학부
2008-11552 구상준

November 17, 2010

1 들어가며

"Hello, world!" 프로그램은 매우 유명한 프로그램이다. 한글로 "안녕, 세상아!" 라고 쓸 수 있는 이 프로그램은 말 그대로 "안녕, 세상아!"를 출력시키는 프로그램이다. 이 프로그램은 거의 모든 프로그래밍 책 1장 1절에 나와 있는 내용이며, 따라서 프로그래밍에 관심 있는 사람에게는 매우 친숙한 프로그램이다.

그러면 여기서 한 가지 질문을 해보자. '얼마나 많은 언어로 "안녕,세상아!" 프로그램을 짜보셨습니까?'라는 질문이다. 자신만만하게 손가락을 꼽으며 'C/C++, Java, ...' 하며 세어보지만 프로그래밍에 익숙한 사람들도 5개를 넘지 못한다. 이상한 프로그래밍 언어에 관심이 있다고 자부하는 나도 12개가 한계다.

이제 wikipedia(`http://en.wikibooks.org/wiki/Computer_Programming/Hello_world`)에 들어가보자. 그러면 200개가 넘는 언어들(!)로 짠 "안녕, 세상아!" 프로그램들을 모두 볼 수 있다. 이 목록을 보노라면 '아! 이 세상에는 정말 많은 프로그래밍 언어가 존재하는구나!' 라고 절로 감탄하게 된다. 그리고 자연스럽게 자문하게 된다. '이 많은 언어들은 어떻게 만들어졌을까?'

답은 간단하다. 많은 프로그래밍 언어들은 '필요하기 때문에' 만들어졌다. 예를 들어, C는 시스템 소프트웨어를 작성하기 위해 만들어졌으며, Ada는 미 국방성에서 군용 소프트웨어를 작성하기 위해서 만들어졌다. 물론 개인의 흥미를 충족시키기 위한 소박한 욕구에서 비롯한 언어들도 있지만 결국 어떤 필요로부터 발명되었다는 사실은 변하지 않는다.

그렇다면, 우리는 어떤 언어를 사용해야할까? 그리고 어떤 언어가 좋은 언어란 말인가? 몇 가지 사례를 들어가며 프로그래밍 언어에 대한 나의 대

답을 담백하게 말하고자 한다.

2 Lisp의 역습

폴 그라함(Paul Graham)은 수필가이자, 프로그래머다. 그는 자신의 에세이를 통해서 자신이 프로그래밍 언어에 대해 어떻게 생각하는지 말하고 있다. 비록 그가 100% 옳지는 않겠지만, 프로그래머들과 경영자들이 편협한 사고를 가지고 있다는 주장에 대해서는 생각해 볼만하다. 그는 어떻게 자신이 Yahoo! Store를 성공적으로 운영했는지를 예로 들고 있다.

그와 로버트 모리스(Robert Morris)는 인터넷 사용자들에게 개인 상점을 열 수 있는 프로그램을 개발해야 했다. 문제는 인터넷 시장이 다른 오프라인 시장과는 비교도 안될정도로 경쟁이 심하다는 데에 있었다. 즉, 남들과는 다른 장점을 가지지 않은 상품은 이 시장에서 매우 빠르게 사장되었다.

그들은 다음과 같은 사실들에 주목했다.

1. 웹 어플리케이션의 경우, '어떤' 프로그램을 사용해도 상관없다. 사용자들은 오직 모니터에 띄워지는 결과만을 확인하지 서버에서 어떤 프로그램을 사용해서 작업을 수행하는지는 상관하지 않는다.

2. 빠른 개발 속도는 이 시장에서 생명이다. 사용자들은 그들이 원하는 바를 바로 반영해줄 수 있는 어플리케이션을 선택하기 때문이다.

따라서 그들은 다른 프로그래머들처럼 C를 사용하는 길을 포기했다. 대신에 그들은 Lisp을 사용했다. C와는 달리 Lisp은 자동으로 메모리 재활용(garbage collection)을 지원해주었으며, 인터프리터로 소스 코드를 바로 시험해볼 수도 있었다. (기타 다른 장점들도 그의 에세이에 자세하게 나와있지만, 간략하게만 소개하는 바다.)

결과는 성공적이었다. 사용자들은 그들의 프로그램을 보고 마치 자신들의 컴퓨터에서 돌리는 듯한 인상을 받았다. 절박해진 경쟁자들은 새로운 기능들을 추가해서 도전했지만, 워낙 개발 속도가 빨랐기 때문에 경쟁자들이 그 기능들을 발표한지 하루나 이틀 사이에 유사한 기능들을 추가할 수 있었다.

그가 생각하기에 다른 경쟁자들이 가진 가장 큰 약점은 편견이었다. 즉, 그들이나 그들의 사장은 C나 Java와 같은 '남들이 흔히 쓰는' 언어에 몹시 익숙해져 있어서 다른 언어들을 사용하는 방식은 생각하지도 못한다는 것이다.

이 고리타분한 생각의 틀을 벗어나서 좋은 언어를 찾아 사용해야 한다는 그의 의견에는 확실히 일리가 있다.

3 Python의 단 맛과 쓴 맛

고백하건대, 나도 하나의 프로그래밍 언어를 신봉하는 '유일신교' 신자였다. 하지만 나는 위의 폴 그라함이 생각한 신자와는 조금 달랐다. 내가 믿었던 언어는 Python(파이썬)이었다.

내가 파이썬을 '믿었던' 이유는 여러 가지가 있다.

- 내가 대학교 1학년 때, 처음 배웠던 언어가 파이썬이었다. 프로그래머가 자신이 처음 배운 언어를 계속 사용하고 의존하는 현상은 흔히 일어나는 바다.
- 문법이 쉽다. main 함수도 필요없고, C나 Java에서 의례적으로 써줘야 하는 세미콜론(;)도 없다.
- 라이브러리 이용이 매우 쉽다. 즉, 다른 사람들이 작성한 프로그램을 이용하기 매우 쉽다. 이미 내 노트북에는 파이썬 과학 라이브러리와 그래프 그리기 라이브러리, 그래픽 처리 라이브러리가 설치되어있다.
- 매우 높은 레벨의 언어다. 내가 C언어를 배울 때에 가장 힘들었었던 것은 문자 포인터가 이상한 곳을 가리켜서 Stack overflow라는 영 반갑지 못한 오류를 받아보았을 때였다. 하지만 파이썬의 경우, String이나 List를 처리할 수 있는 매우 쉬운 방법을 제공하며 오류도 나지 않는다.

이런 이유에서 1,2학년 때 나는 파이썬을 열정적으로 썼었고 주변 사람들에게도 이 좋은 복음(?)을 널리 퍼뜨리려곤 했었다. 비유적으로 말하자면 나는 파이썬의 단 맛에 취해있었던 것이다.

그러던 어느 날, Python에 어두운 모습을 보고 말았다. 내가 PS(Problem solving) 문제를 파이썬으로 풀고 있었을 때의 일이었다. 내 옆에 있었던 친구가 그것을 보더니 자기도 해보겠다고 C로 답안을 작성하기 시작했다. 친구가 답안을 작성하는데는 꽤 시간이 걸렸는데(거의 하루가 걸렸다.), 일단 작성된 답안을 돌리자 1분이 채 못되서 답이 나오는 것이었다. 나는 충격을 받았는데, 같은 알고리즘으로 짠 내 파이썬 코드는 족히 30분은 기다려야 했기 때문이었다.

나중에 동아리 학술제 준비를 하게 되면서, Programming Language Benchmark 사이트를 알게 되었다. (`http://shootout.alioth.debian.org/`) 이 때, 파이썬이 C보다 30배에서 50배는 느리다는 사실을 알 수 있었다. 이 때야 비로소 나는 파이썬의 쓴 맛을 뼈저리게 느꼈다.

4 좋은 언어 잡기

위에서의 사례들을 볼 때, 임의의 프로그램을 짜는데 언제나 가장 좋은 언어는 찾기 힘들다는 사실을 알 수 있다. 이는 우리가 우리 자신에게 끊임없이 다음과 같이 물어보게 한다. '무슨 언어를 여기에서 사용해야 하지?'

물론 좋은 언어와 나쁜 언어가 있으면 좋은 언어를 선택해야 한다. 예를 들어, 폴 그라함이 말한대로 기계어나 다른 저급 언어를 사용하지 않는다. 프로그램을 작성하기도 힘들 뿐더러, 그래야 할 장점이 거의 없기 때문이다.

그렇다면, 어떤 언어가 좋은 언어인가? 이것에 대해서 나는 다음과 같은 언어들이 좋다고 생각한다.

4.1 사용자들이 만들어가는 언어

스틸(Guy Steele)은 좋은 언어가 가져야 할 덕목으로 '사용자들이 쑥쑥 키워나가는 언어'를 들었다. 에세이에서 그는 APL을 예로 들었다. APL은 사랑받지 못하는 언어인데, 사용자들이 타입이나 필요한 구조를 만들어서 그 언어를 확장하는 것이 매우 어렵기 때문이다.

사용자들이 하나씩 참여해서 풍성해지는 언어 (지금에 개발되고 있는 대부분의 고급 언어들이 이런 언어들이다.)가 좀 더 발전되었다고 말할 수 있지 않을까?

4.2 엄밀한 언어

프로그래밍 언어는 정확해야 한다. 내가 프로그래밍 언어 A(A언어을 말하는 것이 아니다.)로 alpha라는 프로그램을 짰다고 가정해보자. 실행기에 따라서 alpha가 주는 결과가 다르다면 얼마나 황당한 일인가? 심지어 실행기에 따라 alpha가 돌기도 하고 돌지 않기도 한다면, 얼마나 당황스러운 일인가? 실제로 C 프로그래밍 숙제를 하면서 이 비슷한 일을 겪었다. 학부 sysprog 실습 서버에서 int가 64비트였기 때문에, 으레 32bit겠거니하고 짜다가 낭패를 본 것이다.

좋은 언어는 컴퓨터에게 명확하게 무엇을 해야할지 지시해줄 수 있어야 하는 언어일 터이다.

4.3 사용자에게 편한 언어

메모리 재활용을 지원하지 않는 언어나, 타입을 점검해 주지 않는 언어를 써 본 사람들은 안다. 얼마나 이런 언어가 불편하고 위험한지말이다. 내가 2학년 때, 컴퓨터 프로그래밍을 들었을 때 메모리 누수를 잡는 프로그램으로 내 프로그램을 점검해보니 free를 제대로 안 해줘서 메모리가 계속해서 새는 현상을 발견했다. 뒤늦게 고치긴 했지만, 숙제이기 망정이지 다른 상용 프로그램이었으면 큰일날 뻔 했다. 드러난 곳이나 숨겨진 곳에서나 사용자에게 편의를 제공해주는 언어가 좀 더 좋다고 말해야 하지 않을까?

5 어떤 언어를 택해야 하지?

그렇지만 이것들을 만족하는 언어는 수도 없이 많고, 또 그렇지 않음에도 쓸만하다고 생각되는 언어들이 많다. 만약 그렇다면 알맞은 언어를 선택하는 기준은 그 언어가 사용되는 작업이라고 말할 수 있다. 앞에서 폴 그라함이 빠른 눈치로 Lisp을 선택해서 성공을 거둔 것이 좋은 예다.

보다 더 자세한 예를 들어보자. 만약 테트리스를 내가 짠다면 무슨 언어가 적당할까? 어떤 언어로도 충분한 성능을 낼 수 있다고 생각되므로 자신이 편한 언어를 사용하면 좋을 것이다. 이제 테트리스를 인터넷 게임 형태로 제공한다고 가정하자. 이 경우에는 짧은 개발시간을 보장하는 언어가 좋으며, Lisp이나 Perl 같은 언어가 적당할 것이다. 3D 테트리스의 경우에는 어떠할까? 3D 게임을 제대로 돌리려면 성능이 우수해야 하며, C나 C++이 이 경우에는 탁월한 선택이 될 것이다.

망치가 아무리 좋은 도구라도 나무를 썰기에는 적절하지 못하지 않겠는가?

6 마무리하며

들어가면서 말했던 “안녕, 세상아!” 프로그램들로 화제를 돌려보자. 웃기지만 진지한 질문을 하나 던져보겠다.

‘무슨 언어가 가장 안녕, 세상아! 프로그램을 잘 짤 수 있을까?’

C나 Java를 생각했다면 여러분은 틀렸다. 정답은 H9Q+다. 이 언어에서 “안녕, 세상아!” 프로그램을 작성하는 방법은 단순히 H라고 쓰는 것이다. 이 언어는 그야말로 “안녕, 세상아!” 프로그램을 작성하기 위해 특화된 언어라고 말할 수 있다.

따라서 우수한 프로그래머 그리고 전산학도라면 다양한 언어를 접하는 것이 필요하다. 이를 통해서

1. 한 언어에 맹목적인 태도를 취하는 것을 피해야 한다.

2. ‘왜 이 언어가 좋은가?’에 대해서 끊임없이 물어야 한다.

3. 그리고 ‘왜 이 언어가 나쁜지?’에 대해서 끊임없이 물어야 한다.

이를 통해서 우리는 조금 더 깨인 시각을 가질 수 있을 것이다. 그때야 한발짝 나아간 모습으로 프로그래밍 세상에 이렇게 외칠 수 있을 것이다. “안녕, 세상아! 내가 간다!”