

학과 공부로 2% 부족한 것

pcpenpal@sparcs.kaist.ac.kr 박용수
SPARCS

개요

- ▶ 공부를 열심히 하고, 추가로 다음에 유의합시다
 - 과목에서 배우지만 실천이 잘 안 되는 것
 - 과목으로 배우기 쉽지 않아 찾아서 메워야 하는 것

- ▶ 그 중 일부인 4가지를 살펴봅니다
 - 프로그래밍 언어
 - 관습
 - 도구
 - 자유/오픈소스 소프트웨어

프로그래밍 언어

들리는 이야기

- ▶ “나는 PL을 왜 들어야 되는지 모르겠어요.”
 - PL 듣는 한 학우
- ▶ “이거 짤 때 뭐로 해야 될까요?”
 - 구현 언어가 명시되지 않은 프로젝트에 고민하는 학우
- ▶ “전 C가 제일 편하던데...”
 - 위 학우의 뒤이은 한 마디
- ▶ “그런 건 어떻게 다 아세요?”
 - 찾으려면 나오는 거 물어본 후 신기해하는 학우

프로그래밍 언어

- ▶ 쓸 줄 아는 프로그래밍 언어엔 무엇이 있습니까?
- ▶ 가장 잘 쓰는 프로그래밍 언어는 무엇입니까?

프로그래밍 언어

- ▶ 언어를 가리지 말아야 한다?
 - 어떤 언어를 사용하는지는 크게 중요한 게 아니다?
 - 장인은 연장을 가리지 않는다?
- ▶ 우리는 어떻게 언어를 공부하는가?
 - CS101
 - 그리고...?
- ▶ 주로 쓰는 언어는?
 - C
 - Java

프로그래밍 언어

▶ 언어 사용의 문제

- 대부분의 명령 중심 언어(imperative language)에 공통으로 있는, 특히 C에도 비슷한게 있는 것만 배우고 씀
- 객체 지향 언어의 경우, 객체 사용의 이점을 누리지 못하는 경우가 대부분
 - 자칫하면 클래스는 네임스페이스로 전략
 - 다형성은 안드로메다로
- C 프로그래밍 시, 배열 외의 다른 자료 구조 사용을 기피
 - 동적 메모리 할당도 기피
 - “그냥 귀찮아서 큰 배열 잡았어요~”
- 언어의 발전을 따라가지 못 함

프로그래밍 언어

- ▶ 막상 제대로 아는/하는 언어가 없다
- ▶ 장인은 연장을 가리지 않는다, 그러나 최적의 연장을 가장 효율적으로 사용한다
- ▶ 주력 언어를 정해 제대로 익히자

제대로 배우기

- ▶ 좋은 바이블 마련하기
 - 책, 웹 문서, 잘 알려진 커뮤니티
- ▶ 기본서를 빈틈 없이 익히기
 - Tip이라고 불리는 것들의 비밀
 - 머리말, 목차, Chapter 1
- ▶ 그 이상 나아가기
 - 예: Effective C++, Effective Java, Code Complete

제대로 배우기

- ▶ 수련하기
 - 일부러 다른 길로 가 보기
- ▶ 탐구하기
 - 예: Java SE API

프로그래밍 언어

- ▶ 다양한 언어를 접해보자
 - 언어가 생각을 지배한다

관습

관습, 좋은 습관

- ▶ 많이 들어도, 못 들어도 본 것
 - 변수 이름 명확히 짓기
 - 들여쓰기 잘 하기
 - 띄어쓰기 잘 하기
- ▶ 지키십니까?
 - 의식적? 무의식적?
- ▶ 못 지키신다면?

관습 convention

- ▶ 쾌적한 삶을 위한 지침 guideline
- ▶ (대부분의 경우) 안 지키다고 큰일나진 않지만, 안 지키면 결국 아쉬운 것
- ▶ 코딩 관습, 문서화 관습, 주석 관습 등
- ▶ 관습이 된 데는 이유가 있다

관습의 예 - 코딩 관습

- ▶ 안 지킨다고 큰일나지 않는다
 - 안 지켜도 조교가 그걸로 점수 깎지 않는다
 - 스펙에 그런 거 지키라고 써 있지도 않다
- ▶ 안 지키면 결국 아쉽다
 - 재수강할 때 저번에 짠 거 열어보면 자기 스스로 피곤
 - 둘이 같이 프로젝트하는데 팀메이트가 힘들어한다

관습

- ▶ 한둘이 아니다
 - 문화의 다양성
- ▶ 스스로 지침을 설계해도 된다
- ▶ 좋은 지침을 골라 따라보는 수련이 우선 필요
 - 아쉽게도 책에서 배우기 쉽지 않음
 - 수업 시간에도 잘 언급되지 않음
 - CS101? CS206?

관습을 넘어

- ▶ CS 집단의 문화와 분위기를 느끼자
- ▶ 모든 것은 사회에서 비롯
- ▶ 전문 용어^{jargon}에도 익숙해지자

도구

도구^{tool}

- ▶ 편집기/IDE는 무엇을 사용하십니까?
 - Vim? Eclipse? Visual Studio?
(Emacs?! TextMate?! UltraEdit?! NetBeans?!)
 - 왜?!
- ▶ 개발 환경 및 컴파일, 빌드 환경 구축에 자신있으십니까?
- ▶ 워드 프로세서의 스타일 기능을 아십니까?

도구

- ▶ 도구를 능숙하게 사용할수록 생산성이 높아진다
- ▶ 도구 사용의 단계
 - 1: 극도로 단순한 업무가 아니면 시도할 엄두를 못 냄
 - 2: 쉬운 것은 해내고, 어려운 건 적절한 근성을 통해서
 - 3: 도구는 나의 수족, 문제의 본질에 집중할 수 있음
- ▶ 노력하지 않으면 2단계에서 멈춤

도구를 접할 때

- ▶ 능동적으로 접하게 되지 못 함
 - “Vim이라는 게 있는데 그걸 써야 됩니다.”
 - Vim을 편집기 그 이상의 무엇으로 착각
 - “그냥 Eclipse 쓰면 돼.”
 - IDE가 편집기로 전략

- ▶ MS 개발 도구를 무심코 쓰는 경우
 - 높은 확률로 우물 안 개구리
 - 불법 복제에 대해 무감각

도구

- ▶ 도구 사용법에 대한 학습의 동기 부여 실패
 - “숙제를 낼 수 있을 만큼만 배우자”
- ▶ 대안에 대해 관심이 없어짐
- ▶ 적절할 때 필요한 도구를 찾아내는 능력 부족
- ▶ 불필요한 근성 발휘하는 잘못된 습관 익힘

도구를 배우자

- ▶ 자주 사용하는 도구는 내 손발 다루듯 익히자
- ▶ 하나의 도구를 잘 익히면 다른 도구 사용법을 습득하는 시간이 빨라짐
- ▶ 언어도 도구다
 - 셸 스크립트
 - 기민한^{agile} (스크립팅) 언어
 - Ruby, Python 등

도구의 효과

- ▶ 도구가 컴퓨터 본질의 이해를 돕는다
 - 컴파일 도구, 빌드 도구
 - 도구의 공부보다 곧 컴퓨터 시스템의 공부
- ▶ 비슷한 여러 도구 경험을 통한 추상화의 발견

도구를 찾아다니자

- ▶ 생각보다 듣기 쉽지 않지만 사실 많이 쓰이는 도구
 - 자동 단위 테스트 도구
 - 코드 버전 관리
 - ...
- ▶ 새로운 도구가 계속 만들어지고 있다
- ▶ 우리도 만들 수 있다

자유 / 오픈소스 소프트웨어

자유 / 오픈소스 소프트웨어

- ▶ 평소 사용하던 소프트웨어를 생각해보자
 - Vim, GCC, G++, Make, Linux, bash, SSH, PuTTY, Mozilla Firefox, Perl, Python, Ruby, MySQL, lex, yacc, Java*, ...
- ▶ 이런 소프트웨어는 우리가 어디서 났을까?
 - 땅 파서?

자유 / 오픈소스 소프트웨어

- ▶ MS Windows 사용자 여러분께
 - 상용 소프트웨어인 건 알고 계시죠?
- ▶ 우린 앞 장의 소프트웨어를 구입해 본 적이 없다

자유 / 오픈소스 소프트웨어

▶ 개요

- 사용, 복사, 수정, (재)배포 등이 자유로움 (제한이 없음)
- 소스 코드의 공개
- 배타적 소유 및 독점의 거부, 카피레프트^{copyleft}
- 원칙과 사상이 담겨있다
 - 자원봉사자 아님
- 프리웨어^{freeware}와 다름

자유 / 오픈소스 소프트웨어

- ▶ 매력적!
 - 멋진 사상
 - 실질적으로 세상의 발전에 지대한 공헌

- ▶ 우리에게 직접적으로 많은 영향
 - 유용한 도구 및 라이브러리로써
 - 공부 - 코드 리딩 - 의 대상으로써
 - 활동할 수 있는 공간으로써

자유 / 오픈소스 소프트웨어

- ▶ 전세계 대학생들의 오픈소스 활동이 활발
 - ICPC가 다가 아니다!
- ▶ 개발만이 활동인 것은 아니다
 - 사용 및 피드백 제공 등도 아주 중요
- ▶ 이런 쪽으로 관심 있는 & 밀어주시는 교수님 부족
 - ICPC는 TCLab의 노력으로 현재 분위기 상승세

자유 / 오픈소스 소프트웨어

- ▶ 폭넓은 배움, 다양한 시야
 - 이런 세계도 있다
- ▶ 분명히 인식하고, 마음에 들면 동참하자

마무리

수업시간에 배운 것 되짚어보기

- ▶ 소프트웨어 공학의 원칙
 - Separation of Concerns
 - Modularity
 - Abstraction
 - Generality
 - ...
- ▶ ADT와 그의 자료 구조
 - Stack, Queue, Linked List, Set, Map, ...
- ▶ 좋은 것 많이 배우긴 하였으나...

CS를 대하는 우리의 자세

- ▶ 더 잘할 수 있지만, 여러 가지 이유로 세상과 타협하고 스스로 만족스럽지 못한 결과물을 내놓는다
 - 스파게티 코드
 - 무조건 정적 배열
 - 추상화 없음
- ▶ 원칙을 지키고 능동적으로 CS를 익혀나간다면?

끝

질문 받습니다