|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 32주차 | **기간** | 2025.01.28  ~2025.02.03 | **지도교수** |  |
| 이번주 한일 요약 | 프로그래머스 도서 첫째마당 00~04 내용 정리 | | | | |

프로그래머스 첫째마당 (00~05)  
00 코딩테스트 준비 전

1.프로그래머스 장점

- 타인의 풀이를 볼 수 있다=> 문제풀이 접근 방식, 코딩 스킬 습득(알고리즘, 입출력, 예외처리 등) => 사고를 넓힐 수 있다.

- 내가 생각한 테스트 케이스를 추가할 수 있다.

- 문제에 있는 테스트케이스에만 잘 동작하도록 코드를 짜지 않고 자신만의 테스트 케이스 추가(권장).

2. 아는 것과 모르는 것을 명확하게

2-1. 기록하라 - 문제를 풀지 못했다면 어디까지 생각해봤는지 기록해두기 => 답안과 비교하며 복기

2-2. 실전처럼 - 자체적으로 시간을 제한하여 주어진 시간을 효율적으로 사용

2-3. 요약하라 - 이해한 뒤 그 내용을 요약하기

01. 코딩테스트 준비

1-1. 언어는 중요하지 않음. 변수선언, 함수 정의, 컨테이너 자료형 다루기, 조건문, 반복문 활용

=> C++ 의 장점 : STL에서 제공하는 다양한 기능

1-2. 문제 분석 연습 : 코딩능력보다는 문제풀이능력이 핵심 => 분석에도 충분한 시간 사용

- 문제를 동작단위로 쪼개서 분석 => 한 번에 생각하는 양 줄이기

- 제약사항 파악 후 테스트케이스 추가 => 어떤 알고리즘을 사용할지, 예외 처리등 에 도움

- 입력 크기 분석 - ex) 100만개의 입력을 O(N^2)로는 시간내에 X. 이후 3장에서 자세히

- 핵심 키워드 파악 : 빠르게 문제를 파악하고 좋은 알고리즘 선택

ex) '최적의 해' => 너비 우선 탐색 고려 / '정렬된 상태의 데이터' => 이진탐색, 파라메트릭 탐색 고려 / '최단 경로' => 다익스트라, 벨만-포드, 플로이드-워셜 알고리즘 고려

- 데이터 흐름, 구성 파악 :

-ex) 삽입과 삭제가 빈번하게 일어날 것 같다 => 힙 자료구조 고려

전화번호부 작성 : 보통 이름으로 전화번호를 검색 - 키는 이름, 값은 전화번호로

1-3. 의사코드 설계 연습

원칙1 : 프로그래밍언어로 작성하면 안 됨

원칙2 : 일반인도 이해할 수 있는 자연어로 작성

원칙3 : 일정한 형식X(자유롭게 작성)

=> 추상 단계에서 설계 진행

- 세부구현이 아닌 동작 중심(프로그래밍 요소X)

- 문제 해결 순서로 작성

- 충분한 테스트 - 구현단계로 갈수록 잘못된 부분 수정비용 커짐

02. 프로그래머스 활용

2-1. 프로그래머스 이전에는 LeetCode등 에서 대비. - 다른환경, 실질적인 학습자료x

프로그래머스 : 많은 IT기업이 코테 진행. 기출문제 확인

2-2. 가이드

- 스킬체크 : 레벨1~5까지의 스킬 키워드 ex) 레벨1 : 문자열, 구현, 정렬, 해시, 그리디

- 레벨 당 정확성/효율성 결과, 점수 => 정확성과 효율성 둘 다 통과하여야 함.

2-3. 문제 풀이 전

컴파일 옵션 - C++버전 확인. 현재 C++17 => 비주얼 스튜디오에도 적용 or 온라인 컴파일러 활용 www.onlinegdb.com

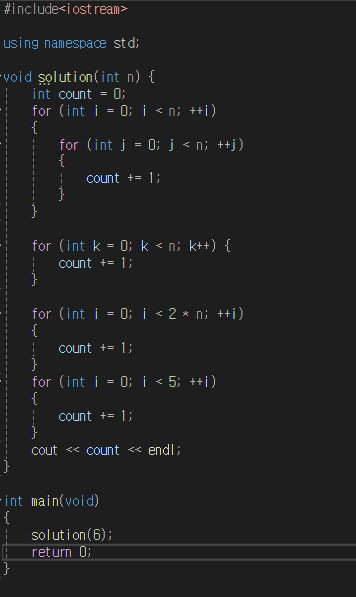
프로토 타입 - solution()함수의 매개변수 타입, 반환 타입 확인

03. 알고리즘 효율 분석

3-1. 시간 복잡도.

- 문제들 마다 '가장 효율적으로 해결하는 알고리즘' - 수행 시간과 관련   
=> 시간 복잡도. 알고리즘의 성능을 나타내는 지표, 입력크기에 따른 연산 횟수.  
 - 빅오 표기법 - 상한선 활용. ex) 3x^2 + 5x + 6 => O(x^2)과 같다.

코드 예시



위 코드의 각 반복문의 연산은 n^2, n, 2n, 5번이다. => f(x) = x^2+3x+5 => O(x^2)  
- 최고차항만 남기고 계수를 지우는 이유 => 상한의 정확한 값X, 정도를 파악하는 추이  
- 코딩테스트에 시간복잡도 활용 => 제한시간 내에 출력값이 나올 수 있는 지(초당 1~3천만)

04. 코딩테스트 필수 문법

4-1. 빌트인 데이터 타입

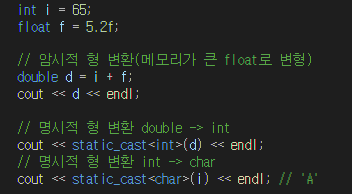
- C++에서 많이 사용하는 빌트인 데이터 타입(헤더 없이 사용가능한 기본 내장 데이터 타입) : int, float, double, bool, char

- 정수형 : 양의 정수, 음의 정수, 0. 사칙연산, 비교연산, 비트연산

- 부동소수형 : float(4바이트-7자리), double(8바이트-15자리). 소수를 저장할 때 사용

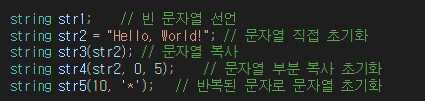
- 형 변환 : 서로다른 변수간 연산을 하여 발생하는 암시적 형 변환 / 임의로 변경하는 명시적 형 변환

예시 코드

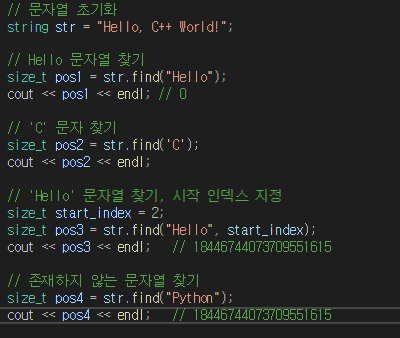


- 문자열 : string 표준 헤더를 추가하여 사용.

- 문자열 선언 및 초기화



- 문자열 찾기 : 특정 문자나 문자열 찾을 때 - find(찾을 문자열) / find(찾을 문자열, 탐색 시작 위치)  
 문자열을 찾으면 해당 문자열이 시작하는 인덱스 반환, 찾지 못하면 string::npos



|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | | |
| **해결방안** |  | | |
| **다음주차** | 33주차 | **다음기간** | 2025.02.04  ~2025.02.10 |
| **다음주 할일** |  | | |
| **지도 교수**  **Comment** |  | | |