

과제 #3

HW1, HW2, HW3, HW4

과제 제출 마감: 10/12 24:00

조교 노인우, inwoo13@hanyang.ac.kr

조교 한중수, soohan@hanyang.ac.kr

Homework

- ◆ 과제 제출 마감: 10/12 24:00
- ◆ Gitlab repository에 “HW3” 폴더를 만든 후 진행
- ◆ 실습 서버 제출, 개인 PC 제출 중 편한 방법으로 제출
- ◆ 각 과제마다 HW3 폴더 내에 과제별 폴더를 만든 후 제출
(e.g. 첫번째 과제의 경우 /HW3/minimal_fighter/ 내에 담아서 제출)
- ◆ 채점 기준은 실습서버 환경에서 채점
- ◆ 모든 skeleton code들은 아래의 링크에서 다운로드 가능
<https://drive.google.com/open?id=0BzwzEKRERCb7U1RLSupIM2JOMWc>

Homework_01 – “Minimal Fighter”

◆ MinimalFighter class 작성

■ private 멤버 변수

int mHp (체력)

int mPower (공격력)

FighterStatus mStatus (상태: Invalid, Alive, Dead – defined by enum)

■ 생성자

MinimalFighter(): 체력 0, 공격력 0, 상태 Invalid로 초기화

MinimalFighter(int _hp, int _power): 체력과 공격력을 주어진 값으로, 상태 Alive 또는 Dead로 초기화 (입력한 체력에 따라 상태가 바뀜)

■ 멤버 함수

int hp() const

int power() const

FighterStatus status() const: 체력, 공격력, 상태 리턴

void setHP(int _hp): 체력을 변경

void hit(MinimalFighter *_enemy): Hit 명령 수행 (규칙 참조)

void attack(MinimalFighter *_target): Attack 명령 수행 (규칙 참조)

void fight(MinimalFighter *_enemy): Fight 명령 수행 (규칙 참조)

Homework_01 – “Minimal Fighter”

◆ 작성 규칙

- 주어진 skeleton code에서 MinimalFighter 클래스를 구현한다.
(인터페이스, 메인함수 부분은 수정하지 않음)
- 규칙을 모두 반영할 수 있도록 구현한다.
- 올바르지 않은 입력이 들어온 경우 종료
(아래 예제의 q 와 같은 규칙 외의 입력들)

◆ 파일명: **minimal_fighter** 폴더 내에 다음과 같은 파일들 존재 (minimal_fighter.cc, minimal_fighter.h, minimal_fighter_main.cc)

- ◆ 입력: 양의 정수나 0인 Fighter1의 체력 및 공격력, 액션(H, A, F), 양의 정수나 0인 Fighter2의 체력 및 공격력
- ◆ 출력: 액션 이후 두 Fighter의 상태를 출력
(fight의 경우 최종 결과만 출력)

Homework_01 – “Minimal Fighter”

◆ Minimal Fighter 교전 규칙

1. 모든 Fighter는 체력이 0 이하로 떨어지면 죽으며 (status = Dead), 입력된 체력이 0 이하일 경우는 이미 죽어있는 상태라 아래의 모든 행동을 할 수 없다.
2. Hit: Fighter가 enemy와 공격을 동시에 한 번씩 교환한다. 공격을 하면 공격력만큼 상대방의 체력이 감소된다.
3. Attack: Fighter가 target을 일방적으로 공격한다. Target의 체력이 공격력만큼 감소하며, Fighter의 공격력이 0이 된다. Fighter의 체력은 변화가 없다.
4. Fight: Fighter와 enemy가 둘 중 하나가 죽을 때까지 공격을 교환한다. 동시에 죽을 경우 둘 다 죽은 것으로 처리한다.
5. 예시
H2, P1 hit H1, P1 → H1, P1 / DEAD
H3, P1 hit H3, P1 → H2, P1 / H2, P1
H1, P3 attack H3, P1 → H1, P0 / DEAD
H4, P1 fight H3, P1 → H3, P1 / H2, P1 → H3, P1 / H2, P1 → H2, P1 / H1, P1 → H1, P1 / DEAD

Homework_01 – “Minimal Fighter”

◆ 입력 및 출력 예시

```
$ ./minimal_fighter
2 1 H 1 1
H1, P1 / DEAD
3 1 H 3 1
H2, P1 / H2, P1
1 3 A 3 1
H1, P0 / DEAD
4 1 F 3 1
H1, P1 / DEAD
q
```

Fighter1 (HP2, POWER1) Hit Fighter2 (HP1, POWER1)

Fighter1 (HP1, POWER1) / Fighter2 는 죽음

규칙 외의 입력으로 종료

Homework_02 – 댓글 관리 프로그램(Reply Administrator) 2

◆ 지난번 만든 댓글 관리 프로그램의 기능이 포함된 ReplyAdmin class 작성

■ private 멤버 변수

string* chats: 댓글 목록

■ 생성자, 소멸자

ReplyAdmin(): chats를 NUM_OF_CHAT 만큼 초기화

~ReplyAdmin(): chats를 delete

■ 멤버 함수

int getChatCount(): 지난 과제의 skeleton code 참조

bool addChat(string _chat): _chat을 chats에 추가 (추가 실패 시 false 리턴)

bool removeChat(int _index): index에 있는 chat 삭제 (_index 없을 시 false 리턴)

bool removeChat(int *_index, int _count): _count 크기의 indices 배열 안에 있는 index에 해당되는 chat을 모두 삭제 (하나라도 삭제 성공했을 시 true 리턴, index가 없을 시 무시)

bool removeChat(int _start, int _end): _start부터 _end까지 chat을 모두 삭제 (하나라도 삭제 성공했을 시 true 리턴, start가 음수거나 end가 chats 보다 클 경우 해당되는 부분만 삭제)

◆ 파일명: reply_admin 폴더 내에 다음과 같은 파일들 존재

(reply_admin.h, reply_admin.cc, reply_admin_main.cc)

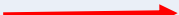

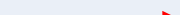
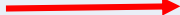
◆ 입출력 및 자세한 기능은 지난 과제의 댓글 관리 프로그램과 동일

Homework_03 – “정수 집합 연산(integer set operations)”

- ◆ 입력 받은 정수 집합에 대해 합집합, 교집합, 차집합을 계산하는 SimpleIntSet class 작성
 - 주어진 simpleIntSet (simple_int_set.h) 클래스를 구현하고 이를 이용한다.
 - 메인함수와 멤버함수의 인터페이스는 변경할 수 없다.
 - 집합의 크기는 MAX_SIZE를 넘지 않는다고 가정한다.
 - 주어진 모든 public 멤버함수를 구현한다.
 - 집합 내의 숫자를 오름차순으로 정렬하여 관리한다.
(지난 과제에서 구현한 정렬 재사용 가능)
 - 한 집합 내에 동일한 원소는 존재할 수 없다. (e.g. { 1 1 2 3 } 입력시 { 1 2 3 }으로 처리)
 - 입력에 오류가 있으면 종료한다.
- ◆ 파일명: simple_int_set 폴더 내에 다음과 같은 파일들 존재
(simple_int_set.h, simple_int_set.cc, simple_int_set_main.cc)
- ◆ 입력: { num1 num2 ... numk1 } OP { num1 num2 ... numk2 }
(단 OP = +, *, -)
- ◆ 출력: 연산 결과 집합을 같은 형식으로 출력

Homework_03 – “정수 집합 연산(integer set operations)”

◆ 입력 및 출력 예시

```
$ ./simple_int_set  
{ 1 2 3 } + { 3 4 5 }  두 집합의 합집합  
{ 1 2 3 4 5 }  
{ -1 5 3 2 } - { 1 2 3 }  두 집합의 차집합  
{ -1 5 }  
{ -1 5 3 2 } * { 1 2 3 }  두 집합의 교집합  
{ 2 3 }  
0  주어진 형식 외의 입력: 종료
```

Homework_04– “이진 탐색 (binary search)”

- ◆ 앞의 SimpleIntSet 내의 원소를 빠르게 찾을 수 있는 이진 탐색 함수 작성
 - 이진 탐색 알고리즘은 아래의 링크 참조
http://en.wikipedia.org/wiki/Binary_search_algorithm
 - 주어진 헤더파일에 있는 BinarySearch 클래스를 구현한다.
 - 해당 원소의 위치를 리턴한다. 주어진 값이 없으면 -1을 리턴한다.
 - 찾을 값이 -999가 입력되면 종료
- ◆ 파일명: **binary_search** 폴더 내에 다음과 같은 파일들 존재
(binary_search.h, binary_search.cc, binary_search_main.cc)
- ◆ 입력: SimpleIntSet 입력 이후 탐색하고자 하는 원소 입력
- ◆ 출력: 해당 탐색 값의 위치

Homework_04– “이진 탐색 (binary search)”

◆ 입력 및 출력 예시

입력: 파란 글씨
출력: 빨간 글씨

```
$ ./binary_search
{ 2 3 5 7 9 }
2
0
5
2
4
-1
-3
-1
100
-1
9
4
-999
```

→ -999 입력: 종료

