더 좋은 코드 짜기

SSL

좋은 코드란?

1. 좋은 코드는 이해하기 쉬워야한다.

2. 다른 사람이 이해하는데 들이는 시간을 최소화해야한다.

3. 분량이 적다고 항상 좋은 것은 아니다.

1.

```
for (Node* node = list->head; node != NULL; node = node->next)
        Print(node->data);
```

2.

```
Node* node = list->head;
if (node == NULL) return;

While(node->next != NULL) {

Print(node->data);
node = node->next;

}
if (node != NULL) Print(node->data);
```

SSL

1.
```
return exponent >= 0 ? mantissa * (1 << exponent) :  mantissa / (1 << -exponent);
```

2.
```
if (exponent >= 0) {
        return mantissa * (1 << exponent);
}
else {
        return mantissa / (1 << -exponent);
}
```

표면적 수준에서의 개선

1. 특정한 단어를 골라라.

(1) def GetPage(url):

(2) class BinaryTree {

Int Size();

};

(3) class Thread {

void Stop();

};

특정 의미를 내포하는 단어로 변경

단어	대안
send	deliver, dispatch, announce, distribute
find	search, extract, locate, recover
start	launch, create, begin, open
make	create, set up, build, generate, add, new

```
private int level = 5; //난이도 (총알 개수 결정) 처음 시작은 easy로
private int hozzi_x = 200, hozzi_y = 180;              // 호찌 시작 위치
private int hozzi_size_x = 25, hozzi_size_y = 25;     // 호찌 크기
private int field_size_x = 400 , field_size_y = 400; // 필드 크기
private int bullet_size = 10;
private double size = 0;                    // 총알 크기
```

↓

```
private int level = 5; //난이도 (총알 개수 결정) 처음 시작은 easy로
private int hozzi_x = 200, hozzi_y = 180;              // 호찌 시작 위치
private int hozzi_size_x = 25, hozzi_size_y = 25;     // 호찌 크기
private int field_size_x = 400 , field_size_y = 400; // 필드 크기
private int bullet_number = 10;
private double bullet_size = 0;
private int i, score = 0, bonus = 1; // 점수
```

SSL

```c
void Makepuzzle()
{
    int num;
    int width;  // 퍼즐의 N 크기

    int psize = width*width; // 퍼즐 전체 사이즈

    int *puzzle = (int*)calloc(psize, sizeof(width));
    for (int num = 0; num < psize; num++)
        puzzle[num] = num+1;
```

↓

```c
void Makepuzzle()
{
    int num;
    int width;  // 퍼즐의 N 크기

    int puzzle_size = width*width; // 퍼즐 전체 사이즈

    int *puzzle = (int*)calloc(puzzle_size, sizeof(width));
    for (int num = 0; num < puzzle_size; num++)
        puzzle[num] = num+1;

    Initpuzzle(puzzle, puzzle_size, width); //퍼즐의 초기화 ( 맞출 수 있는 경우만 초기화함)
```

SSL

2. 보편적인 이름은 피해라.

```
int Euclidean_norm(float* v, int times) {
int retval = 0.0;
for (int i = 0; i < times; i++)
retval += v[i] * v[i];

return sqrt(retval);
}
```

→

```
int Euclidean_norm(float* v, int times) {
int sum_squares = 0.0;
for (int i = 0; i < times; i++)
sum_squares += v[i] * v[i];

return sqrt(sum_squares);
}
```

Temp의 사용

1.
```
if (right < left) {
tmp = right;
right = left;
left tmp;
}
```

2.
```
string user_info = user.name(); // tmp가 아닌 user_info
user_info += " " + user.phone_number();
user_info += " " + user.email();
```

루프반복자의 사용

```
for (int i = 0; i < clubs.size(); i++)
        for (int j = 0; j < clubs[i].members.size(); j++)
                for (int k = 0; k < users.size(); k++)
                        if (!strcmp(clubs[i].members[k], users[j]))
                                printf("유저는 클럽에 존재합니다.₩n");
```

↓

```
for (int ci = 0; ci < clubs.size(); ci++)
        for (int mi = 0; j < clubs[ci].members.size(); mi++)
                for (int ui = 0; ui < users.size(); ui++)
                        if (!strcmp(clubs[ci].members[mi], users[ui]))
                                printf("유저는 클럽에 존재합니다.₩n");
```

단위를 포함하는 값

```
if (num == 1)
{
    start = clock();
    Makepuzzle();
    end = clock();
    res = (float)(end - start) / CLOCKS_PER_SEC;
```

↓

```
if (num == 1)
{
    start_ms = clock();
    Makepuzzle();
    end_ms = clock();
    res = (float)(end_ms - start_ms) / CLOCKS_PER_SEC;
```

SSL

인수 단위를 포함하게 변경

함수	인수 단위를 포함하게 재작성
Start(int delay)	delay -> delay_secs
CreateCache(int size)	size -> size_mb
ThrottleDownload(float limit)	limit -> max_kbps
Rotate(float angle)	angle -> degrees_cw

이름의 길이

=>사용 범위가 넓으면 긴 이름을 사용하라.

여러 페이지에 걸쳐서 사용되는 변수의 이름을 짧은 문자로 구성해 의미를 알아보기 힘들게 지어선 안된다.

이름 포맷팅으로 의미 전달

```
static const int kMaxOpenFiles = 100;

class LogReader {
public :
        void OpenFile(string local_file);
private :
        int offset_;
        DISALLOW_COPY_AND_ASSIGN(LogReader);
};
```

3. 오해의 소지가 있는 이름은 쓰지마라.

```
result = Database.all_object.filter("year <= 2011");
```

Result는 year <= 2011인 객체들인가?
year <= 2011이 아닌 객체들인가?

=> 고르는 기능 : select()
제거하는 기능 : exclude()

SSL

```
// 텍스트의 끝을 오려낸 다음 '...' 을 붙인다
def Clip(text, length): ...
```

Clip()은 문단의 끝에서부터 length만큼 오려내는 것인가?
문단의 처음에서 length만큼 오려내는 것인가?

Length는 최대 length를 말하는 것인가? 아니면 고정 length를 말하는 것인가?

⇒ 처음에서 length만큼 오리는 경우 : Truncate()
최대 length를 뜻했을 경우 : max_length

한계값 및 범위 이름

first last

1	2	3	4	5	

begin end

1	2	3	4	5	

경계를 포함하는 한계값을 다룰 때는 min, max를 사용

경계를 포함하는 범위에는 first와 last 를 사용

경계를 포함하고 끝에는 배제하는 범위에서는 begin와 end를 사용

Boolean 변수 이름

```
boolean read_password = true;
```

read_password는 패스워드를 읽어야한다는 뜻인가?
패스워드를 이미 읽었다는 뜻인가?

⇒ boolean연산자는 is, has, can, should와 같은 단어를 사용한다.

4. 미학적인 코드가 사용하기 더 편하다.

- 일관성 있는 레이아웃을 사용하라.
- 비슷한 코드는 서로 비슷해 보이게 만들어라.
- 서로 연관된 코드는 하나의 블록으로 묶어라.

일관성과 간결성을 위해서 줄 바꿈

```
switch (mesg) {
case WM_CREATE:
    hWnd2 = CreateWindow("WND2",
        "자식윈도우",
        WS_OVERLAPPED | WS_VISIBLE,
        2, 398, 590, 200,
        hWnd, NULL, _hInstance, NULL);

    hStatic = CreateWindow("STATIC",
        "",
        WS_VISIBLE | WS_CHILD,
        300, 30, 290, 260,
        hWnd, NULL, _hInstance, NULL);

    hEdt = CreateWindow(
        "EDIT",
        "",
        WS_CHILD | WS_VISIBLE |ES_AUTOHSCROLL | ES_AUTOVSCROLL | ES_MULTILINE,
        0, 0, 300, 350,
        hWnd, (HMENU)888, _hInstance,
        NULL
    );

    break;
```

```
switch (mesg) {
case WM_CREATE:
    hWnd2 = CreateWindow(
        "WND2",
        "자식윈도우",
        WS_OVERLAPPED | WS_VISIBLE,
        2, 398, 590, 200,
        hWnd, NULL, _hInstance, NULL);

    hStatic = CreateWindow(
        "STATIC",
        "",
        WS_VISIBLE | WS_CHILD,
        300, 30, 290, 260,
        hWnd, NULL, _hInstance, NULL);

    hEdt = CreateWindow(
        "EDIT",
        "",
        WS_CHILD | WS_VISIBLE |ES_AUTOHSCROLL | ES_AUTOVSCROLL | ES_MULTILINE,
        0, 0, 300, 350,
        hWnd, (HMENU)888, _hInstance, NULL);

    break;
```

SSL

선언문을 블록으로 구성

```
class FrontendServer {
public :
FrontendServer();
void ViewProfile(HttpRequest* request);
void OpenDatabase(string location, string user);
void SaveProfile(HttpRequest* request);
string ExtractQueryParam(HttpRequest* requet, string param);
void ReplyOK(HttpRequest* request, string html);
void FindFriends(HttpRequest* request, string html);
void ReplyNotFound(HttpRequest* request, string error);
void CloseDatabase(string location);
~FrontendServer();
};
```

```
class FrontendServer {
public :
FrontendServer();
~FrontendServer();

// 핸들러들
void ViewProfile(HttpRequest* request);
void SaveProfile(HttpRequest* request);
void FindFriends(HttpRequest* request, string html);

// 질의/응답 유틸리티
string ExtractQueryParam(HttpRequest* requet, string
param);
void ReplyOK(HttpRequest* request, string html);
void ReplyNotFound(HttpRequest* request, string error);

// 데이터베이스 헬퍼들
void OpenDatabase(string location, string user);
void CloseDatabase(string location);

};
```

개인적인 스타일과 일관성

- 클래스, 메소드에서 중괄호 같은 열에 넣기
- 빈 괄호는 생략
- 처음이 아닌 마지막에 쉼표
- snake_case 보다 camelCase 선호
- 큰따옴표보다는 작은따옴표
- 상수 이름은 모두 대문자로
- 탭보다는 스페이스
- 라인길이는 최대 80자

=> 일관성 있는 스타일은 중요하다

주석

1. 코드에서 빠르게 유추할 수 잇는 내용은 주석으로 달지 않는다.

```
class Account {
public :
//생성자
Account();

//profit에 새로운 값 설정
void SetProfit(double profit);

// 이 어카운트의 profit을 반환
double GetProfit();
};
```

2. 변명을 하지말고 이름을 고쳐라

```
private void button1_Click(object sender, EventArgs e)
{
    init(); // 게임 시작 전 초기화
    this.button1.Visible = false; //스타트 버튼 숨김
    this.label2.Visible = false; // label 숨김
    this.label3.Visible = false;
    this.label5.Visible = false;

    gameflag = 1;
    Cursor.Current = Cursors.Default;
    Cursor.Hide(); //커서 숨김


    loading(sender, e); //로딩
    timer2.Start();
}
```

SSL

3. 코드에 있는 결함을 설명하라

함수	인수 단위를 포함하게 재작성
TODO	아직 하지 않은 일
FIXME	오동작을 일으킨다고 알려진 코드
HACK	아름답지 않는 해결책
XXX	여기에 큰 위험이 있다

4. 상수에 대해 설명하라

NUM_THREADS = 8 // 이 상수값이 2 * num_processors보다 크거나 같으면 된다.

---

endfile.priority = 5; // 우선순위가 4가 최대이므로 5를 종료조건으로 주었다.

```
for (int i = 0; i < TN; i++)
{
sem_wait(&emptysem); // wait(empty) bounded buffer가 비어있으면 기다린다.
sem_wait(&closesem); // wait(mutex)

insert_min_heap(&boundbuffer, &endfile); // Critical Section

 //Exit section
sem_post(&closesem); // signal(mutex)
sem_post(&fullsem); //signal(full)
}
```

5. 쉽게 빠질 것 같은 함정을 경고하라.

```
//외부 서비스를 호출하여 이메일 서비스를 호출한다(1분 이후 타임아웃된다.)
void SendEmail(char* to, char* subject, char* body);
```

6. 신중하게 고른 예로 주석을 서술하라.

```
// 예: Strip("abba/a/ba", "ab")은 "/a/"를 반환한다.
String Strip(String src, String chars) {}
```

7. 나올 것 같은 질문을 예측한다.

8. 대명사가 여러 가지를 가리킬 경우에는 사용하지 않는다.

//데이터를 캐시에 넣어라.하지만 데이터가 너무 큰지 먼저 확인하라.

9. 함수의 동작을 실제로 할 수 있는 한도 내에서 최대한 명확하게 설명하라.

```
// 파일 안에 새 줄을 나타내는 바이트('\n')가 몇 개 있는지 샌다.
int CountLines(String filename) {}
```

읽기 좋은 흐름제어짜기

1. 조건문에서 인수의 순서

```
if(length >= 10)

while (bytes_received < bytes_expected)
```

왼쪽	오른쪽
값이 더 유동적인 '질문을 받는' 표현	고정적인 값, 비교대상으로 사용되는 표현

2. if/else 블록의 순서

```
if (a != b) {
// 두 번째 경우
}
else {
// 첫 번째 경우
}
```

```
if (a == b) {
// 첫 번째 경우
}
else {
// 두 번째 경우
}
```

부정이 아닌 긍정을 선호한다.
하지만 특정 경우가 간단하거나 확실하면 먼저 처리하라.

3. 삼항 연산자 ?:는 매우 간단할 때만 사용한다.

4. do/while 그리고 goto 구조는 가독성을 떨어뜨리므로 되도록 사용하지 않는다.

5. 함수 내부의 중첩 제거

```
for (int i = 0; i < results.size(); i++) {
if (results[i] != NULL) {
non_null_count++;
if (result[i]->name != "") {
printf("실행중...₩n");
}
}
```

→

```
for (int i = 0; i < results.size(); i++) {
if (results[i] != NULL) continue;
non_null_count++;
if (results[i]->name == "") continue;
printf("실행중...₩n");
}
```

참고문헌

1] 읽기 좋은 코드가 좋은 코드다
(저자 : 더스틴 보즈웰, 트레버 파우커)

2] http://www.itworld.co.kr

3] https://en.wikipedia.org/wiki/Programming_style