# Coding Rules

## Objectives

- 오류 없는 프로그램
- 이해하기 쉬운 코드 → 유지보수가 쉬운 코드

## Guide Line

- Indent는 Tab이 아닌 Space로, 2또는 4 ( 4를 권장 )
- Function, Method는 한 화면 안에 들어오도록 (20~30줄 내외)
- brace는 새로운 줄에서 시작

```
if ( condition )
{
  // .. do something
}
```

- 2단계를 초과하는 brace는 새로운 method, function으로 작성
- 변수의 초기화에 주의
- recursive에 주의
- heap에 할당하는 메모리의 해제에 관한 해제 확인 ( GC가 있는 경우 제외 )
- 변수의 이름은 hunggarian notation을 참고한 naming
  - static; s_ 접두어
  - global; g_ 접두어
  - member; m_ 접두어
  - string; strName, strAddress, ...
  - int; nID, ...
  - resource; IDC_BTN_, IDC_EDIT, IDC_TREECTRL, ...
  - controls; m_btnOK, m_editAddress, m_treectrlDirectory, ...
- function, method의 naming rule 은 V(동사)+N(명사)를 기본으로 확장

```
bool GetPropertyWithName( string strName )
{
  // .. do something
}
```

- 같은 프로젝트 내에서 같은 function, method 이름 회피
- 조건문의 경우 가능하다면 '상수 (조건문) 변수'의 형태

```
if ( NULL == pszAddress )
{
```

```
  // .. do something
}
```

- 리턴값의 확인; 리턴 값이 있는 함수의 호출의 경우 가능하면 리턴 값을 확인한다.
- while, goto 사용 자제; 코드의 가독성을 위해 for 문이나 foreach를 권장한다.
- 전역 변수의 사용 자제; 특별한 경우가 아니라면 전역 변수는 사용하지 않기를 권장한다.
- 상수는 한 곳에 정의; 숫자를 직접 사용하기 보다 상수를 정의하여 사용하고, 상수는 하나의 파일에 정의하여 모아두거나 해당 소스의 가장 처음에 정의해둔다.

# Coding Conventions

| Coding Conventions | | | |
|---|---|---|---|
| **name** | **description** | **Usually used** | **examples** |
| flatcase | | package at Java, etc. | |
| kebab-case<br>lisp-case<br>spinal-case<br>caterpillar-case<br>dash-case<br>hyphen-case<br>css-case | only small cases, spread with hypens | css, | kebab-case, lisp-case, spinal-case |
| camelCase | | variables at Java, C#, etc. | let myVariable: string |
| PascalCase<br>CapitalCamelCase | | Class | class MyClass |
| snake_case<br>c_case | | variables at Python, PHP, c, etc. | snake_case |
| UPPER_CASE_SNAKE_CASE<br>(UPPER_CASE)<br>MACRO_CASE | | | UPPER_CASE_SNAKE_CASE |
| COBOL-CASE<br>TRAIN-CASE | | | COBOL-CASE |

- Most Common Programming Case Types