CODE COMPLETE 스터디
32.5 주석 작성 기법

컨텐츠검색플랫폼팀
조영일

목차

- line comment
- paragraph comment
- data declaration comment
- control statement comment
- routine comment
- class/file/program comment

한 줄마다 주석 작성하기

- 언제 사용하지?
  - 보통은 필요없다
  - 복잡하거나 오류가 발생하여 기록을 남기고 싶을 때만

한 줄마다 주석 작성하기

- 줄 끝 주석(endline comment)은 피할 것!
  - 오른쪽 정렬이 어려우니 스타일을 맞추기 쉽지 않음
  - 길이 조절이 어려움
  - 코드보다 의미있는 주석을 작성하기 어려움

The comments merely repeat the code.

```
memoryToInitialize = MemoryAvailable();     // get amount of memory available
pointer = GetMemory( memoryToInitialize ); // get a ptr to the available memory
ZeroMemory( pointer, memoryToInitialize ); // set memory to 0
...
FreeMemory( pointer );                      // free memory allocated
```

  - 블럭 주석을 줄 끝 주석으로 작성하지 말 것

```
For rateIdx = 1 to rateCount                    ' Compute discounted rates
   LookupRegularRate( rateIdx, regularRate )
   rate( rateIdx ) = regularRate * discount( rateIdx )
Next
```

  - 유지 보수 기록을 남기지 말 것

한 줄마다 주석 작성하기

- 사용해도 되는 상황
- 데이터 선언할 때

```
int boundary;        // upper index of sorted part of array
String insertVal;    // data elmt to insert in sorted part of array
int insertPos;       // position to insert elmt in sorted part of array
```

- 블럭의 끝을 표시할 때

코드 단락에 대한 주석 작성하기

- 코드의 목적 수준에서

```
/* check each character in "inputString" until a dollar sign
is found or all characters have been checked
*/
done = False;
maxLen = inputString.length();
i = 0;
while ( !done && ( i < maxLen ) ) {
   if ( inputString[ i ] == '$' ) {
      done = True;
   }
   else {
      i++;
   }
}
```

코드 단락에 대한 주석 작성하기

- 문서화에 들일 노력을 코드에

Here's the variable that contains the result of the search.

```
// find the command-word terminator
foundTheTerminator = False;
maxCommandLength = inputString.length();
testCharPosition = 0;
while ( !foundTheTerminator && ( testCharPosition < maxCommandLength ) ) {
   if ( inputString[ testCharPosition ] == COMMAND_WORD_TERMINATOR ) {
      foundTheTerminator = True;
      terminatorPosition = testCharPosition;
   }
   else {
      testCharPosition = testCharPosition + 1;
   }
}
```

코드 단락에 대한 주석 작성하기

- 방법보다는 이유를
  - 방법

```
// if account flag is zero
if ( accountFlag == 0 ) ...
```

  - 이유

```
// if establishing a new account
if ( accountFlag == 0 ) ...
```

- 훌륭한 버전

```
// if establishing a new account
if ( accountType == AccountType.NewAccount ) ...
```

  - 사실 주석이 필요없음
  - 요약 수준의 주석으로 변경

```
// establish a new account
if ( accountType == AccountType.NewAccount ) {
   ...
}
```

코드 단락에 대한 주석 작성하기

- 독자에게 다음에 올 내용을 알려주기 위해
- 주석은 적당량만
- 일상적이지 않은 내용을

```
for ( element = 0; element < elementCount; element++ ) {
   // Use right shift to divide by two. Substituting the
   // right-shift operation cuts the loop time by 75%.
   elementList[ element ] = elementList[ element ] >> 1;
}
```

  - shift 연산(>>)을 나누기 연산(/) 대신에 사용하였음
- 생략하지 말 것
  - 읽기 쉽게

코드 단락에 대한 주석 작성하기

- 메이저 주석과 마이너 주석을 구분
  - 밑줄을 긋거나 ...을 이용

The major comment is underlined.

```
// copy the string portion of the table, along the way omitting
// strings that are to be deleted
//-------------------------------------------------------------
```

A minor comment that is part of the action described by the major comment isn't underlined here...

```
// determine number of strings in the table
...
```

...or here.

```
// mark the strings to be deleted
```

The major comment is formatted normally.

```
// copy the string portion of the table, along the way omitting
// strings that are to be deleted
```

A minor comment that is part of the action described by the major comment is preceded by an ellipsis here...

```
// ... determine number of strings in the table
...
```

...and here.

```
// ... mark the strings to be deleted
```

코드 단락에 대한 주석 작성하기

- 언어/개발환경에서의 오류나 문서에 없는 기능에 대해

```
blockSize = optimalBlockSize( numItems, sizePerItem );

/* The following code is necessary to work around an error in
WriteData() that appears only when the third parameter
equals 500. '500' has been replaced with a named constant
for clarity.
*/
if ( blockSize == WRITEDATA_BROKEN_SIZE ) {
   blockSize = WRITEDATA_WORKAROUND_SIZE;
}
WriteData ( file, data, blockSize );
```

- 좋은 프로그래밍 방식을 위배한 것에 대해

- 교묘한 코드는 주석을 달지 말고 재작성할 것

```
// die a horrible death.
// VERY IMPORTANT NOTE:
// The constructor for this class takes a reference to a UiPublication.
// The UiPublication object MUST NOT BE DESTROYED before the DatabasePublication
// object. If it is, the DatabasePublication object will cause the program to
```

데이터 선언에 주석 작성하기

- 데이터 사용하는 곳보다는 선언하는 곳이 유용함
- 수치 데이터 단위
- 허용 가능한 값의 범위
- 코드의 의미

```
Dim cursorX As Integer  ' horizontal cursor position; ranges from 1..MaxCols
Dim cursorY As Integer  ' vertical cursor position; ranges from 1..MaxRows

Dim antennaLength As Long      ' length of antenna in meters; range is >= 2
Dim signalStrength As Integer  ' strength of signal in kilowatts; range is >= 1

Dim characterCode As Integer      ' ASCII character code; ranges from 0..255
Dim characterAttribute As Integer ' 0=Plain; 1=Italic; 2=Bold; 3=BoldItalic
Dim characterSize As Integer      ' size of character in points; ranges from 4..127
```

데이터 선언에 주석 작성하기

- 입력 데이터의 한계
- 비트 수준의 플래그

```
' The meanings of the bits in StatusFlags are as follows:
' MSB   0     error detected: 1=yes, 0=no
'       1-2   kind of error: 0=syntax, 1=warning, 2=severe, 3=fatal
'       3     reserved (should be 0)
'       4     printer status: 1=ready, 0=not ready
'       ...
'       14    not used (should be 0)
' LSB   15-32 not used (should be 0)
Dim StatusFlags As Integer
```

- 변수와 연관된 주석은 변수 이름을 적어둘 것
- 전역 데이터

제어 구조에 주석 작성하기

- 제어 구조 앞(위)에

Purpose of the following loop

```
// copy input field up to comma
while ( ( *inputString != ',' ) && ( *inputString != END_OF_STRING ) ) {
   *field = *inputString;
   field++;
   inputString++;
} // while -- copy input field

*field = END_OF_STRING;

// if at end of string, all actions are complete
if ( *inputString != END_OF_STRING ) {
   // read past comma and subsequent blanks to get to the next input field
   inputString++;
   while ( ( *inputString == ' ' ) && ( *inputString != END_OF_STRING ) ) {
      inputString++;
   }
} // if -- at end of string
```

End of the loop (useful for longer, nested loops—although the need for such a comment indicates overly

Purpose of the conditional

Purpose of the loop. Position of comment makes it clear that inputString *is being set up for the loop.*

- if, case, loop, block 앞에
- 각 제어 구조 끝에 무엇이 끝났는지를
  - 그러나 이것을 경고 신호로 인식해야 함

루틴에 주석 달기

- 루틴 앞에 위치한 주석은 나쁨
  - 함수 코드 크기보다 주석 크기가 더 클 수 있음
  - 루틴의 "목적", "알고리즘"이 이름보다 더 나은 설명이 되기 어려움
  - 유지보수가 어려움
  - 주석 작성의 부담이 리팩토링을 꺼리게 만듬

```
'*****************************************************************************
' Name: CopyString
'
' Purpose:      This routine copies a string from the source
'               string (source) to the target string (target).
'
' Algorithm:    It gets the length of "source" and then copies each
'               character, one at a time, into "target". It uses
'               the loop index as an array index into both "source"
'               and "target" and increments the loop/array index
'               after each character is copied.
'
' Inputs:       input    The string to be copied
'
' Outputs:      output   The string to receive the copy of "input"
'
' Interface Assumptions: None
'
' Modification History: None
'
' Author:       Dwight K. Coder
' Date Created: 10/1/04
' Phone:        (555) 222-2255
' SSN:          111-22-3333
' Eye Color:    Green
' Maiden Name:  None
' Blood Type:   AB-
' Mother's Maiden Name: None
' Favorite Car: Pontiac Aztek
' Personalized License Plate: "Tek-ie"
'*****************************************************************************
```

루틴에 주석달기

- 설명하고자 하는 코드 가까이
- 루틴 윗부분에 한두 문장으로
- 매개변수가 선언된 곳에

```
public void InsertionSort(
   int[] dataToSort, // elements to sort in locations firstElement..lastElement
   int firstElement, // index of first element to sort (>=0)
   int lastElement // index of last element to sort (<= MAX_ELEMENTS)
)
```

- 유틸리티 활용
  - Javadoc, doxygen 등
- 입력과 출력 데이터 구분

```
void StringCopy(
   char *target,     // out: string to copy to
   char *source      // in: string to copy from
)
...
```

루틴에 주석달기

- 인터페이스 가정
- 루틴의 한계
- 루틴이 미치는 전역적인 효과
- 사용된 알고리즘의 출처
- 프로그램의 위치 표시

클래스와 파일, 그리고 프로그램에 대한 주석

- 클래스 문서 지침
- 파일 문서 지침
- 프로그램 문서 지침

클래스와 파일, 그리고 프로그램에 대한 주석

- 책 패러다임
  - 코드를 책으로 생각
  - 서문, 목차, 단락, 상호 참조,
  - 효과
    - 1000줄짜리 프로그램
    - 경험있는 직업 프로그래머들
    - 유지보수 작업의 평균 시간이 3/4로 단축
    - 유지보수 점수는 20% 향상
    - 코드 이해도는 10~20% 향상

THE END