continue, break, goto 명령문에 대한 고찰

Sunny Kwak

(sunnykwak@hanmail.net)

페이스북 생활코딩에 올라온 질문

- “반복문에서 continue를 거의 사용 안하는 편인가요? 제가 그런 얘기를 들어서....” 이라는 질문이 제기 되었습니다.

- 제 답변 중에서는 이런 말도 있습니다.
  “출처가 불분명한 얘기는 무시하시는 게 정신건강에 좋다고 생각합니다. 제 정신(?) 박힌 guru들은 레퍼런스 링크(reference link)를 제공하거나 소스, 벤치마크 결과를 함께 보여줍니다.”

- 하지만, 별로 중요하지 않은 듯한 질문이라도...
  답변하시는 분들이 ‘고수’라면 심도 있는 통찰을 얻게 되기도 합니다.

성능 측면에서 본 continue, break 그리고 goto

```
while ( /* .. */ ) {
    while ( /* .. */ ) {
       /* .. */ break;
    }
    here
    /* when the break executes,
       the program continues to execute from here */
}

while ( /* .. */ ) {
    /* when the continue executes,
       the program continues to execute from here */
here while ( /* .. */ ) {
       if( /* .. */ ) contiune;
    }
}

{
    goto here;
    /* .. */

    here: /* define the label 'here' */
here /* .. */
}
```

continue는 사람에게는 친절한 가이드가 되고,
machine 입장에서는 아무런 성능 저하가 없습니다.

break 이나 continue는 그냥 goto의 다른 표현이라서...

while block {} 이 정상적으로 끝나고 다시 돌아가
작업하는 것도 내부적으로 goto가 동작합니다.

break, continue, goto 명령문은 조심히 쓰자.

- continue 를 자주 쓰는 언어가 있고 자주 안쓰는 언어가 있다. 당연히 goto와 마찬가지로 사용하는 사람이 어떻게 쓰느냐에 따라 양날의 검이 된다.
  다만, 요즘은 대부분의 언어가 컬렉션(리스트, 배열, 집합, 사전 등)에 필터, 맵등의 고차함수를 사용할 수 있어서 continue를 직접 사용하는 경우가 많이 없다.
  (오현석 의견)

- 더글라스 클락포드는 javascript에서는 continue를 쓰지 않는게 좋고 대부분 사용 안하게 변경 할 수 있다고 하던데 .. 언어마다 차이일듯 C++에서는 별로 대안이 없다.
  (신성철 의견)

- coding style guideline 같은 키워드와 조합하면 관련 가이드가 있을 것이다.
  (천경수 의견)

- 결론은 '사용 빈도가 줄었다'는 이야기가 왜곡되서 '쓰지 말아라'로 전해진 듯 하다
  혹은 성능 보다는 '코딩 가이드 라인', '코딩 스타일' 에 대한 지침인 것 같다.
  (곽중선 정리)

자원 관리 측면에서 접근해보면...

- continue를 쓸 때는 주의가 많이 필요한 편이다. Continue 이후의 구문에서 continue 이전에 할당받은 리소스를 해제하는 것이 있다면 신경써서 생략되지 않도록 해줘야 한다.
  (엄윤섭 의견)

- 부연하자면, Java 같은 가상 머신(Virtual Machine) 기반에서 동작하는 언어에서는 자원 반환을 가상 머신에서 처리하니 별로 신경 안쓰게 되지만, C 혹은 C++ 같은 Native Language 로 개발할 경우에는 조심해야 한다.
  (곽중선 의견)

일반적인 가이드 라인

- goto구문에 대해서는 프로그래밍언어 론에서 상당히 오랫동안 논쟁이 있는 이슈이며, 유명한 교수님들 사이에서도 의견 차이가 존재한다. 다만 전체적으로 공감을 형성한 것은 goto가 많으면 프로그램의 맥락을 파악하기 어렵다는 것이다. 그래서 특정한 경우의 장점을 기술한 글이 있다.

- 자바는 그래서 goto 구문을 제공하지 않는다. 마찬가지로 break, continue도 많이 사용하면 맥락 파악이 어렵다는 것은 널리 공감되고 있다. 특히 실수하기 쉽고, 실수한 부분을 발견하기 어렵다는 것이다. 다만 억지로 break, continue를 안쓰도록 바꾸니 더 읽기 어려운 코드가 되는 경우가 있으므로 대개의 언어가 goto 기능을 제공한다.

- 간단한 경우에는 if 블럭으로 감싸서 continue를 대체할 수 있다.
  다만 이러한 것이 복잡해지면 중첩 if 문과 block이 복잡해져서 오히려 가독성이 떨어진다.

```
------------------------------------------------------------
if (expr)
   continue;
   ...
------------------------------------------------------------
if (!expr)
{
   ...
}
------------------------------------------------------------
```

(금동철 의견)

그리고, 한 말씀 더...

- 자연스럽게 읽히는 코드를 쓰는데 집중하고 개별 규칙에 너무 메이지 않는게 중요한 원칙이다. 물론 초식을 잊어도되는 고수가 되려면 피나는 노력이 필요하긴 하다. (오현석 의견)

- 읽기 쉬운 코드가 절대선은 아니다. 일반적으로 언어를 배울 때는 읽기 쉬운 코드로 단련하지만, 어느 레벨을 넘어가면 굳이 읽기 쉬운 코드에 집착할 필요는 없어진다.
  커널(kernel) 코드나 DBMS 같은 경우를 보면 읽기 쉬운 코드와는 거리가 먼 경우도 많다. 그리고 과거에 규제되던 프로그래밍 원리나 기법 중 몇몇은 시대가 바뀌면서 그 의미가 퇴색된 경우도 많다. (김선영 의견)

Thanks to...

- 페이스북 생활코딩 그룹에서 지혜를 나누어 주시는 분들께 감사 드립니다.
- 모든 분의 의견을 모으기 보다는 초급 개발자가 꼭 알아두면 좋을 것 같은 의견들만 추려 봤습니다.
- 본 문서는 페이스북의 특성 상 오래된 토론을 찾기 어렵다는 분들을 위해 정리한 것이며, 저작권 없이 공개합니다.
- 문서에 대한 추가 의견을 얻기 위함과 최초 작성자에 대한 정보 차원에서 제 이메일을 적어 둡니다.