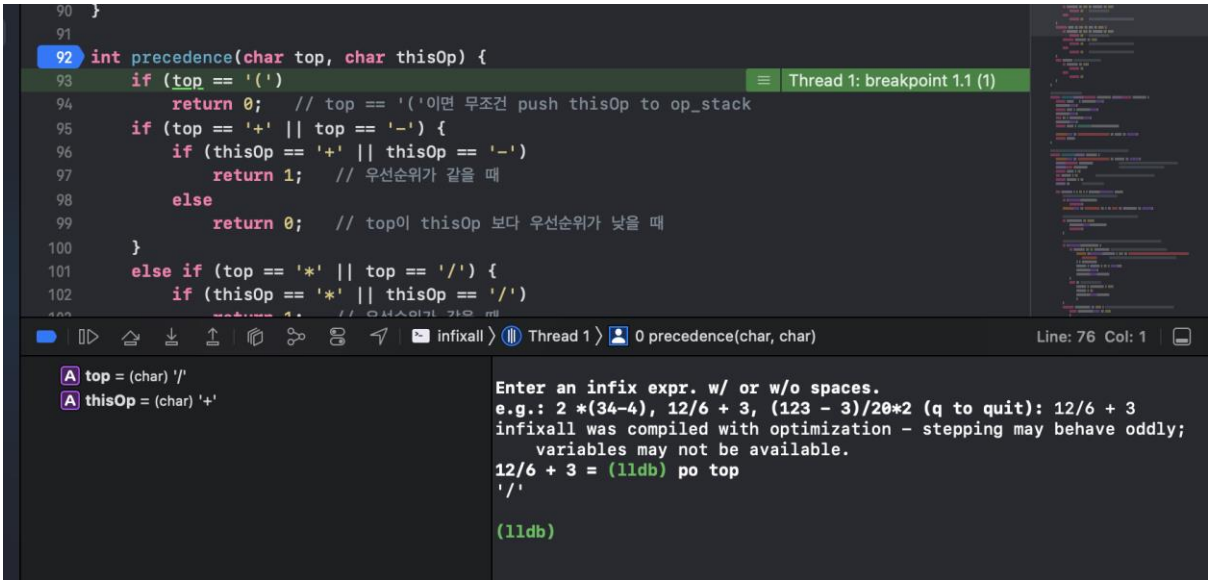
<Using XCode debugger> **22000711 조다빈**

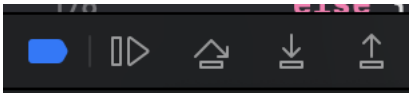
1. po 명령어

- breakpoint를 기점으로 변수에 어떤 값이 저장되어 있는지 알 수 있다
- LLDB에서 po {변수}를 입력하면 변수에 대한 description이 출력된다
- 런타임 때 변수값을 직접 수정할 수도 있다
- p 를 사용하여 primitive type 변수 또는 구조체 변수 값을 확인할 수도 있다
  ex. po top = '+'
- 느낀점 : 파일을 실행하던 중 오류를 발견하여 변수의 값을 바꾸고 결과를 바로 확인하고 싶을 때, 굳이 다시 코딩하여 컴파일 하지 않아도 po 명령어를 사용하여 바로 변경된 결과값을 볼 수 있다는 점이 편리했다.
- 사용 예시 : prcedence를 사용하여 연산자 우선순위를 확인할 때 오류가 생겨서, precedence 함수에 break를 걸어 argument가 제대로 보내졌는지 확인을 했다.



2. Breakpoint

- 코드 왼쪽 코드 라인을 클릭하면 breakpoint를 생성할 수 있다
- 이렇게 생성한 breakpoint를 line breakpoint라고 부른다. 코드 라인 그 자체에 breakpoint를 생성하는 것이다.
- breakpoint를 제거하려면, 생성한 breakpoint를 드래그해서 밖으로 끄집어내면 된다.
- 왼쪽부터 순서대로 :



1. Deactivate breakpoints
   Break Point 활성화/비활성화
2. Continue program execution
   Break Point 무시하고 다시 실행
3. Step over
   Break Point가 걸린 곳에서 다음 한 줄만 실행

4. Step into
   Break Point가 걸린 곳에서 함수 내부로 들어가기
5. Step out
   Break Point가 걸린 곳에서 함수 외부로 들어가기

- 느낀점 : for(i = 0…){..} 에서 i 가 특정 값일 때 결과가 잘못 나오면 line breakpoint를 설정하여 step into 또는 step over 을 사용해 내가 예상한 대로 함수를 수행하는지 확인할 수 있었다. 또 Variable View에 breakpoint 가 설정된 타이밍에 변수 값이 나와있어서 정말 편했다. 특히 이번 과제에서 stack에 올바른 값이 들어갔는지 확인해야하는 경우가 많았는데, 이렇게 container를 사용하는 코딩에 breakpoint를 사용하면 유용하겠군 이라고 생각했다.
- 사용 예시 : tokens[i] == '(' 일 때, 예상한 것과 다른 결과값이 나와서 148 line에 breakpoint를 걸었다. 확인해보니 1번 if 문만 수행해야 하는데 2번 else 문도 수행하는 것이다. 알고보니 1번 if문에 continue; 를 넣지 않아서 2번 else 문도 수행했던 것이다. 그래서 1번 if 문에 continue; 를 추가해줬다.

```
146
147         // token is an opening brace, push to op_stack
148         if (tokens[i] == '('){
149             op_stack.push(tokens[i]);
150             continue;
151         }
152
153         // current token is a value(or operand), push it to va_stack.
154         if (isdigit(tokens[i])) {
155             if (pdigit == 1) {          // 그 전 index 문자가 숫자일 때 = 여러자리 숫자일 때
156                 assert (!isspace(tokens[i - 1]) && "There should be no space between t
                        numbers");      // 1 23 + 2 를 입력한 경우 123으로 인식하지 않고 경고문 출력
157                 s = tokens[i];
158                 ivalue = ivalue * 10 + stod(s);
159                 va_stack.pop();
160                 va_stack.push(ivalue);
161             }
162             else {  // 한 자리 숫자일 때
163                 ivalue = tokens[i] - '0';
164                 pdigit = 1;
165                 va_stack.push(ivalue);
166                 }
167         }
168         else if (tokens[i] == ')') { // compute it, push the result to va_stack
169             while (op_stack.top() != '(')
170             {
171                 assert (va_stack.size() >= 2);
172                 pdigit = 0;
173                 value = compute(va_stack, op_stack);
174                 va_stack.push(value);
175             }
176             op_stack.pop();
177         }
178         else { // token is an operator; push it to op_stack.
179             pdigit = 0;
180             char thisOp = tokens[i];
181             while (op_stack.size() != 0 && precedence(op_stack.top(), thisOp) >= 1)
182             {
183                 assert (va_stack.size() >= 2);
```

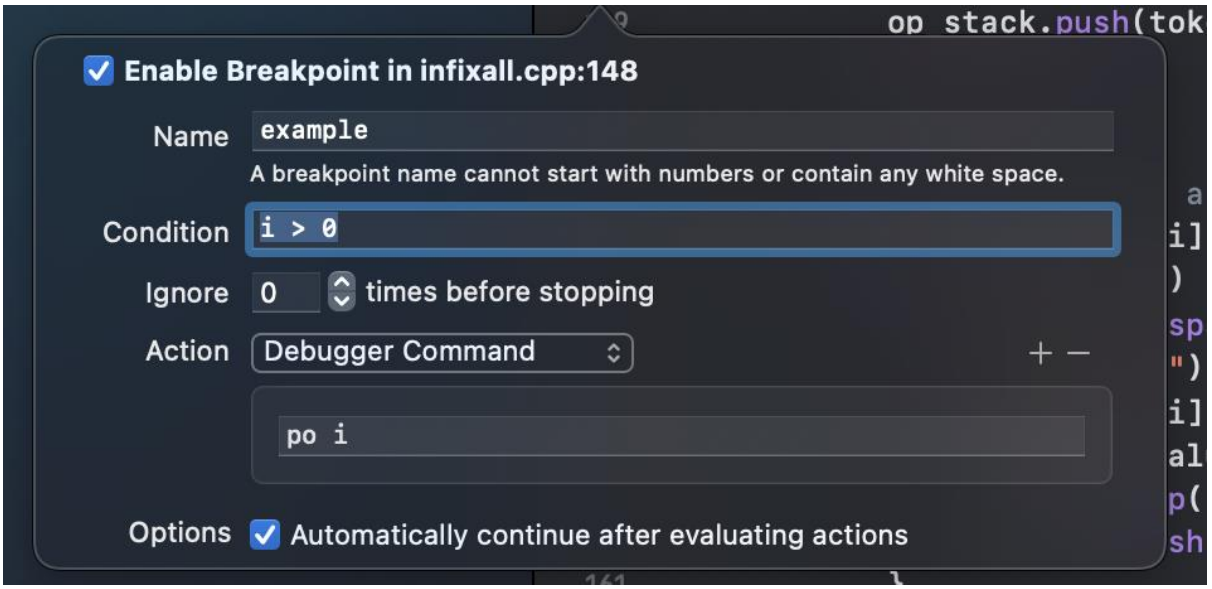
- Edit Breakpoint :
  - 생성한 breakpoint를 더블 클릭하면 Edit Breakpoint 창이 나온다. (breakpoint 우클릭 >

Edit Breakpoint를 눌러도 같은 효과)

- Name : breakpoint의 name을 지정할 수 있다
- Condition : 특정 조건일 때 break가 걸린다. (swift 문법으로 작성한다)
- Ignore : 특정 횟수 이후부터 break가 걸린다
- Action : break가 걸리기 전, 특정 동작을 수행한다. (LLDB 명령어, script, Sound 등)
- Automatically continue after evaluating actions : Edit Breakpoint에 설정한 동작은 수행하지만, break가 걸리지 않는다

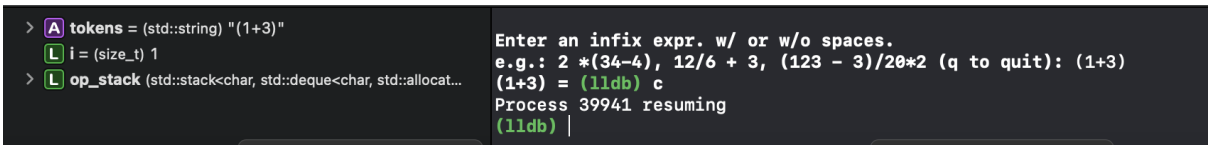
Ex)

- example 이란 breakpoint는 i > 0 일 때 action을 수행한다
- action은 디버그 콘솔창에 i 를 출력하는 것이고,
- Automatically continue after evaluating actions 가 체크되었기 때문에, action 수행 후 break 가 걸리지 않는다.



3. c 명령어

- c 명령어를 통해 breakpoint를 떠나서 계속 진행(continue) 할 수 있다
- 느낀점 :
- 사용 예시 : (1+3)이 stack에 들어가는 과정을 보기 위해 break 를 걸었는데, 예상대로 진행되길래 c 명령어로 사용했다



깨달은 점 : 런타임 오류가 발생했을 때 사용하면 좋은 프로그램이다. 또 이때까지 내가 간단한 프로그램을 작성했을 때는 DPRINT를 사용하여 직접 디버깅을 했지만, 앞으로 .cpp 파일도 많아지고 복잡한 파일을 실행할 때는 XCode는 필수라고 생각한다. 또 디버그 모드 말고 릴리즈 모드라는 것도 알게 되었는데, 파일을 배포할 때와 개발자가 디버깅할 때 두가지 버전을 나눠서 작업할 수 있는 환경을 만들어 놓은 것도 정말 편리했다. 처음엔 release 모드로 하다가 런타임 에러가 발생하면 debug 모드로 전환해서 세세하게 알아볼 수 있는게 정말 편했다