<디버깅 에세이 Debugging Essay>

22200550 이예은

처음에는 Xcode를 사용하는 방법을 몰랐습니다. 그렇다는 이유로 과제를 시작하지 않을 수는 없어서 급한대로 처음엔 visual studio code를 사용해서 컴파일을 했었습니다. visual studio code에서 문제가 일어났을 때는 에러 메시지를 읽어보고 그 함수 주변에서 cout으로 일일이 변수의 값들을 출력하여 확인해보면서 디버깅을 해나갔습니다. 나중에 TA세션에 참여하여 Xcode 사용하는 방법을 정확하게 익히고 infixall.cpp 파일을 Xcode를 이용해서 만들었습니다. 이전에 infix.cpp를 이미 visual studio code를 이용해서 만들어 놓아서 그렇게 많이 바꿀 필요는 없어서 크게 오류나는 부분이 잘 없었습니다. 그래서 visual studio code와는 다르게 Xcode에서만 사용가능한 한줄씩 디버깅하는 기능을 사용하지는 않았습니다. 하지만 infixall이 실행은 됐지만 중간에 실행하다가 오류나는 부분이 있었습니다. 중간에 op_stack이 하나 밖에 남지 않은 상황에서 compute라는 함수를 실행을 하게 되어서 right 는 그 값이 제대로 들어왔지만 left는 이미 stack이 빈 상황이라 값을 받을 수 없는 오류가 발생했습니다. 그냥 visual studio code에서 돌렸더라면 그냥 오류 메시지만 띄웠을 거 같습니다. 아니면 특정한 값을 넣었을 때 오류가 났다면 그 예시를 가지고 일일이 눈으로 코드를 돌려보면서 어디서 문제가 생겼을지 확인해보면서 코드의 오류를 확인 해봤을 거 같습니다. 또 눈으로 돌려서 오류가 거기서 났는지 확인해보기 위해 뭐가 틀렸는지, 내가 예상한 부분에서 오류가 일어난게 진짜 맞는지 알아보려고 cout을 일일이 다 써서 각각의 변수에 어떠한 값이 들어갔는지 확인해봤을 거 같습니다. 하지만 Xcode를 사용했을 때는 right에는 va_stack.top() 값이 들어왔다는 것을 확인 할 수 있었고 left에는 쓰레기 값이 들어가는 것을 확인할 수 있었습니다. 그래서 밑에 빨간색으로 표시한 조건을 추가하여 그러한 상황을 만들지 않게 바로 고칠 수 있었습니다.

```
double evaluate(string tokens) {
    DPRINT(cout << ">evaluate: tokens=" << tokens << endl;);
    stack<double>  va_stack;                // stack to store operands or values
    stack<char> op_stack;                   // stack to store operators.
    double value = 0;

    for (size_t i = 0; i < tokens.length(); i++) {
        // token is a whitespace or an opening brace, skip it.
        if (isspace(tokens[i])) continue;

        DPRINT(cout << " tokens[" << i << "]=" << tokens[i] << endl;);

        // current token is a value(or operand), push it to va_stack.
        if (isdigit(tokens[i])) {
            int count=0;
            double result=0;
            while(isdigit(tokens[i])){
                i++;
                count++;
```

```
            }
            i--;
            for(int k=0; k<count; k++){

                result+=(tokens[i-k]-'0')*pow(10,k);

            }
            va_stack.push(result);


        }
        else if (tokens[i] == ')') { // compute it, push the result to va_stack

            while(op_stack.top() != '('){
                double value = compute(va_stack, op_stack);
                va_stack.push(value);
            }
            op_stack.pop();
            printStack(op_stack);
            //cout << "$$\n";


        }
        else if(tokens[i] == '(') op_stack.push(tokens[i]);
        else { // token is an operator: push it to op_stack.

            while(op_stack.size()!=0 && op_stack.top() != '(' && (precedence(tokens[i]) <=
precedence(op_stack.top()))){
                double value = compute(va_stack, op_stack);
                va_stack.push(value);
            }


            op_stack.push(tokens[i]);
        }
    }

    DPRINT(cout << "tokens exhausted: now, check two stacks:" << endl;);

    DPRINT(printStack(va_stack);  cout << endl;);
    DPRINT(printStack(op_stack);  cout << endl;);

    //cout << "your code here: process if !op_stack.empty() \n";

    while(!(op_stack.empty())){
        value = compute(va_stack, op_stack);
        va_stack.push(value);
    }

    value = va_stack.top();
```

```
    //cout << "your code here: post-conditions\n";
    assert(op_stack.size()==0);
    assert(va_stack.size()==1);

    //cout << "your code here: return & clean-up\n";


    return value;
}
```

Xcode를 사용하는 방법을 좀 더 일찍 알았더라면 infix.cpp와 postfix.cpp 파일 둘 다 cout를 일일이 출력해가면서 변수에 어떤 값이 들어가는지, 이 함수가 어디서 지금 실행되는지, 이 함수가 몇 번까지 돌아가는지 등등을 확인을 빠르게 할 수 있었을 거 같다라는 생각이 들었습니다, 이런 편리한 툴을 미리 알지 못했다는 점이 조금 아쉬웠습니다. 큰 프로젝트를 할 때에는 확실히 visual studio code를 사용하는 거보다 좀 더 어디서 오류가 일어나는지 그 때 그 변수 안에는 어떤 값이 들어가는지 다 알려주는 Xcode가 편리하다는 점을 깨달았습니다.

또 Xcode를 이용할 때 좋았던 점은 visual studio code를 이용했을 때는 syntax error만 알려줬을 거 같습니다. 하지만  Xcode는 여기에서 지금 변수에 아무런 값이 들어가있지 않다라는 오류 메시지를 바로 바로 옆에 띄워주었습니다. syntax error또한 바로 바로 알려주어서 코드 짤 때 그런 것 까지 다 고려해가면서 코드를 짤 수 있어서 그런 점이 또 visual studio code보다 Xcode가 더 나은 점이었습니다.

처음 tool 사용하는 방법은 visual studio code가 Xcode보다 더 간편하고 편리하였습니다. 하지만 시작만 편리할 뿐 오류를 찾으려면 일일이 변수에 어떠한 값이 들어가는지 코드를 짜는 사람이 확인을 했다면 Xcode는 처음 설정만 조금 어려울 뿐 어떤 오류가 일어나는지 더 정확하게 알려주었습니다. 또 코드를 짜는 동안에도 옆에 error 메시지를 보여주어 컴파일하기 전에도 어떤 변수를 놓치는 것을 미리 미리 확인할 수 있어서 코드를 짜는 순간에 바로 바로 코드를 고칠 수 있어서 이정도의 장점이면 처음 설정이 조금 어렵더라도 Xcode를 사용하는 편이 장기적으로 봤을 땐 훨씬 효율적이고 편리할 수 있다고 생각했습니다. 앞으로 visual studio code라는 tool을 사용하기 보다는 Xcode와 같이 미리 미리 오류를 확인할 수 있고 컴파일 다 하고 실행 과정 중에 오류가 나타나더라도 좀 더 오류를 자세히 알려주는 tool을 사용해서 코드를 짜야겠다는 생각을 했습니다.

Apple store에서 Xcode를 다운받으려고 했는데 평점이 너무 좋지 않아서 왜 이렇게까지 좋지 않을까라는 의문이 코드를 짜는 동안 계속 들었습니다. visual studio code 라는 tool을 사용하면서 코드를 짰던 저로서는 Xcode의 이런 디버깅해주는 방식이 너무 좋아서 “왜 이렇게 까지 평점이 나쁠까?” “지금 나에게는 너무 디버깅하기 편리한 tool인데?” 라는 생각을 계속 하게 했습니다. visual studio가 xcode보다 너 나은 것인가 라는 의문이 들면서 다음에 window 노트북을 가지게 된다면 Visual Studio를 이용해서 코드를 짜보고 싶었습니다. 그리고 코드를 짤 때 사용하기 편한 tool을 이용하기 보다는 디버깅하기 편한 tool을 앞으로 사용

해야겠다는 생각이 들었습니다.