言語処理系の開発にあたって

954480 岡本悠里

第1版　2022/01/30

この課題を行って会得したことや感想などを三千字(3,000字)以上

この授業で学んだプログラミング言語は、パラダイムを学ぶものとなった。プログラミング言語には関数、論理などスタイルが決まっている場合が多いと感じた。マルチパラダイム言語でもどのパラダイムを採用するかを決めている。今回の課題で言語を作るとなった時に、新しいパラダイムを考案しなければ新しく言語を作る意味がないと思い、新しいパラダイムを考え出そうとした。言うまでもなく、苦戦し、パラダイムを作り出すとは何なのかについて考えざるを得なくなった。いくつかのパラダイムを眺めているうちに、パラダイムというのは作ろうとして作れるものではないのではないかと考えた。困ることや要望があってそれを解決する手段として考え出されるという流れを踏み、先にパラダイムを作り終えてその後にどういった要望が叶えられるかという順ではないのはないかと思った。つまり、この課題を取り組むには日頃困っていることなどを洗い出し、それらを解決するような言語を作るという流れが自然ではないかと考えた。よって、今回はパラダイムを考え出すことまでは出来なかったが、英数字の羅列が怖いと思う困りごとを想定し、これを解決するような言語を作ろうということになった。更に、自分の経験から値の代入または束縛において1回生の時に感じた「=」を使う違和感について解消しようと考えた。統合できないものに=を使う事に違和感を感じる上に、代入と束縛の違いが全く理解できなかった。視覚的に最初から荷札だと思わせてしまえば良いでのはないかと考えた。つまり、この言語を使うユーザを玄人ではなく素人とした。yaccやlexを用いながら自分がデザインした言語を実装していく体験ができるということはもちろんなのだが、実際に世の中に出た際に誰が使うのだろうかということを考える機会になった。先生の授業を聞いていきなり全部実装するというのはやめて、少しの機能から実装しなさいというアドバイスをもとに最初は定義と足し算引き算しかないものを作った。たかが足し算引き算だが、実装は困難を極めた。この僅かな機能であっても起こりうる想定外の事態がたくさん考えられた。決めた文法に沿って定義が行われるように、その定義を踏まえて正しく計算できるようにするだけでもこの場合はうまくいくべきであり、この場合はエラーにしなければなど不具合の削減が難しい。今まで行ってきたソフトウェア開発でのテストとは違ったテストであると感じた。単体テストや結合テストなどだけでなく、そもそもの言語を作っているので、起こりうる入力パターンを確実に掬い上げる必要がある。テストケースをできるだけ多く作り出し、あらゆる脅威に対応することの重要性と難しさを学んだ。私は最初、lexで文字列の定義をクオート( ‘ )で囲まれたすべての文字を文字列と定義していた。考えられる入力を考えているうちに、クオートが含まれた文字列が渡された時にうまくいかないことに気づいた。文字列でクオートを使いたい時に困ってしまう。たかが定義、たかが文字列、たかがprintだがこのように不具合が発生する。結局、\’と入力しエスケープさせることでクオートも文字列として定義することができるようになった。少ない機能にもかかわらず次々不具合がでてきた。一気にwhileやらforやらifを注ぎ込んで作成していたら処理系を作ることにも至らず、一度も動くことがなかったのではないかと思う。時間や技術の限界で足し算引き算printほどの小さな機能しか織り込めなかったが、プロンプタをターミナルに出し、足し算や引き算を行うことができる言語を作ることができた。