群馬大学大学院工学研究科

情報工学専攻 横内研究室

URL：http://www.cs.gunma-u.ac.jp/~yokouchi/lab/

■研究テーマ
●プログラミング言語の基礎理論
●オブジェクト指向プログラミング言語
●関数型言語

■キーワード
プログラミング言語、Java、オブジェクト指向プログラミング、プログラム検証

■産業界の相談に対応できる技術分野
プログラミング言語関連技術、オブジェクト指向プログラミング関連技術、ソフトウェア検証技術

横内寛文 准教授

連 絡 先
情報工学専攻　横内寛文　TEL：0277-30-1818　FAX：0277-30-1801　e-mail：yokouchi@cs.gunma-u.ac.jp

研究概要

複雑なことがらを簡単に ～プログラミング言語の研究～

◆

私の研究室では、コンピュータのプログラミング言語とその基礎理論について研究しています。コンピュータは大別するとハードウェアとソフトウェアからできています。ハードウェアとはコンピュータ本体を意味し、ソフトウェアはコンピュータをどのように使うかの技術のことをいいます。プログラミング言語はソフトウェアの中核的な技術です。

研究室のゼミ風景

特徴と強み

分かりやすい指令書を書くための ソフトウェア開発の中核的技術

◆◆

コンピュータはプログラムと呼ばれる指令書に従って動作し、この指令書を書くための言語がプログラミング言語です。プログラミング言語に求められることは、複雑なことがらを簡単に記述できることです。コンピュータが直接解釈できる命令は、足し算や掛算といったごく単純なものだけです。しかし、ワープロやインターネット・ブラウザといった複雑なソフトウェアを開発する際には、プログラムの意図が誰にでも分かる形で表現できることが望まれます。そのためには、プログラムのためのより高次な概念や機能が必要になります。プログラミング言語の研究では、コンピュータへの個々の処理を抽象化ないしは一般化して、さまざまなプログラム概念や機能が開発されています。さらには抽象化そのものを表現する手法が考案されたりしています。

具体的な例を用いて、抽象化とか意図を表現するとはどういうことかをみてみます。大学のプログラミング演習の課題でよく使われている問題に「微分・積分プログラム」というものがあります。商用のプログラムにはかなわないとしても、学生でもそれなりに強力なプログラムが書けます。ただし、このプログラムではかなり複雑な処理が必要になります。たとえば、得られた数式を自動的に簡単化するという処理がいります。人間が数式を簡単化するときには、ほとんど無意識のうちに図1のような規則を適用しています。しかし、既存のプログラミング言語でこの規則と同じことをさせようとすると、図2のようなプログラムを書かなくてはなりません。このプログラムが図1の規則とどのように対応しているかは、すぐに理解するのは難しそうです。これに対して、図3は同じプログラムを私の研究室で開発しているプログラミング言語で書いたものです。このプログラミング言語では、データ処理の抽象化のひとつである「パターンマッチ」と呼ばれる機能が使われています。詳しい説明は省略しますが、図1の変形規則がほとんどそのままの形で記述されていることがわかると思います。

```
a + 0          => a
0 + a          => a
a + (b + c)    => (a + b) + c
数値 + 数値    => 数値
数値 + b       => b + 数値
```

図1　数式の簡単化規則の一部

```
if (exp.isAdd()) {
    Exp e = (Add)exp;
    if (e.right().isNum()) {
        Num m = (Num)e.right();
        if (m.val() == 0)
            return e.left();
    }
    else if (e.right().isAdd()) {
        Add e2 = (Add)e.right();
. . . . . .
```

図2　従来言語(Java)による数式の簡単化プログラム

```
match (exp) {
  with (Add(Exp a, Num(0))) { return a; }
  with (Add(Num(0), Exp a)) { return a; }
  with (Add(Exp a, Add(Exp b, Exp c))) {
          return new Add(new Add(a, b), c); }
  with (Add(Num(int m), Num(int n))) {
          return new Num(m + n); }
  with (Add(Num a, Exp b)) {
          return new Add(b, a); }
  . . . . . .
  // (注) Exp 数式、Add 足し算、Num 数値
}
```

図3　筆者の研究室で開発中の言語による数式の簡単化プログラム

今後の展開

プログラムすることなく プログラムを作ることも

◆◆◆

史上初のプログラミング言語は1950年代に開発されたFortranという言語でした。それ以降、星の数ほどのプログラミング言語が考案されています。現在、もっともよく使われているものはC(C++)とJavaと呼ばれる言語です。ほとんどの理工系の大学でもこの2つの言語を教えています。その他にも、とくに理論計算機科学の研究者の間では、関数型言語と呼ばれる先進的な言語が愛用されていて、この分野の言語はプログラミング言語全般の研究にも大きく影響を与えています。図3で紹介したプログラミング言語も、関数型言語の分野での研究成果を、もっと一般的な言語に取り入れるというプロジェクトの一環として開発しています。さらには、図形を使ったりマウス操作を使ったり、あるいは新しい装置を使って、従来とはまったく違った方法でプログラムを作成するような研究も行われています。プログラムすることなくプログラムを作るという意味で、プログラミングレス・プログラミングと呼ばれています。プログラミング言語そのものは、将来にわたってソフトウェア開発の中核的な技術であり続けるものと思われます。