COMS4115 – Programming Languages & Translators

Project Proposal

Group members:
Bernardo Abreu – bd2440
Felipe Rocha – flt2107
Henrique Grando – hp2409
Hugo Sousa – ha2398

Description

DaMPL (Data Manipulation Programming Language) is a language aimed to data manipulation applications. It allows the user to define the way the information is structured and provides types and methods to easily manipulate it. It features read functions in the standard modules that can obtain information from different file extensions, such as ".csv" and ".xml" in a single function call. The language is translated to C and compiled using gcc.

DaMPL (Data Manipulation Programming Language)

```
//This is a comment in DaMPL

//DaMPL has the following bult-in types
integer i = 1;
float f = 1.0;
string s = "aaa";
bool b = true;

//These are boolean expressions
(i > 0) //true
(i is integer) //true
(s = "ab") //false

//DaMPL also has loop structures
for(int i = 0; i < n; i++) {}
while(i < n) {}

//In DaMPL, non-void functions must declare the return
// variable on the prototype
integer result sum(integer a, integer b) {
        result = a+b;
}

//Also, DaMPL has built-in containers called holders
```

```
holder<int> h1;
h1.append(1);
h1.append(2);
h1[0] //returns 1

// Blocks
//To manipulate structured data, DaMPL has blocks
block bb {
        integer a,
  string s
};

// Block instantiation
bb bb1;

//the following assignments are equivalent
bb1.a = 1;
bb1.(0) = 1;

//Type checking
(bb1.a is integer)



// Here's how DaMPL works on its main purpose
block Account {
        int id,
        string name,
        float checking_bal,
        float savings_bal
};

block Compact_Account {
        int id,
        string name,
        float total_bal
};

Compact_Account out compact(Account in) {
        out.id = in.id;
        out.name = in.name;
        out.total_bal = in.checking_bal + in.savings_bal;
}

holder<Account> accounts;
```

```
// To fill holder values from a csv file, we use the CSV module
accounts = CSV:read("in.csv");
// This module calls the following built-in function
// (which could have been called directly
accounts.fromGeneric("in.csv","%0%,%1%,%2%,%3%");
// This can be also written as
accounts.fromGeneric("in.csv","%id%,%name%,%checking_bal%,%savings_bal%");

// Let's convert different block types (basic data manipulation)
holder<Compact_Account> new_accounts = accounts.map(compact);

Account a1 = {20,"Henrique",200,100};
new_accounts.append(compact(a1));


XML:write("out.xml",new_accounts);
// The following call would produce the same result
accounts.toGeneric("out.xml","<entry><id>%0%</id><name>%1%</name><checking_bal>%2%<
/checking_bal>"+
        "<savings_bal>%3%</savings_bal></entry>);

// CSV and XML are modules written in DaMPL !
module CSV {
// (...)
        // read function just create the pattern string depending on the block definition
  // and calls the Holder fromGeneric function. Note that holder is parameterized
        holder<?T> out read(string filename) {
        string patternStr;
   integer n = blocksize(T); //This will give the number of elements in T
   for(integer i=0;i<n;i++) {
        patternStr += "%" + i + "%";
    if(i != n-1) {
                patternStr += ",";
    }
   }
        out.fromGeneric(filename,patternStr);
  }

// (...)
}
```

DaMPL will also support read and write from DBMS, allowing automatic block definition from the DBMS internal schema.