15

An Overview of ‘Swaram’: A Language for Programming in Tamil

S.G. Ganesh

(Software Engineer, Hewlett-Packard, Bangalore, India)

G.R. Prakash and K.K. Ravi Kumar

(Software Engineers, Verizon, Chennai, India)

Introduction:

“Swaram” is a simple, general-purpose and procedural language designed for programming in Tamil. The authors of this paper are the designers and developers of a working version of the compiler and a virtual machine (interpreter) as a ‘proof of concept’. The sample implementation was done as an academic project, when the authors were pursuing their MCA degree in PSG College of Technology, Coimbatore, India. This paper provides a brief overview on motivation for the project, design considerations, technical overview of the implementation of Swaram. Few sample programs are also provided for illustrating the programming aspects of Swaram.

Motivation:

Knowledge of English language is a pre-requisite for computer programming (as of now, at least in India) and computing should not be the luxury of a privileged few, who know English. To enable the common man to work in his mother tongue, programming languages should be available in regional languages as well.

This is a well-recognized need. Providing a preprocessor (the simplest alternative), a full-fledged native-language programming environment (for example, Visual Basic language and environ-ment in French), or even a full-fledged programming language (for instance, the Ruby language for Japanese) are few of the well-known means to achieve the same end.

Two important questions naturally arise here. Which is the best alternative? If a preprocessor will do, isn’t it an overkill to provide a programming language? There is no single and simple answer for these questions. There are many perspectives from which we have to approach this question. Let us answer these questions indirectly by taking-up analogies for providing design and usage perspectives.

Today, printing high-resolution documents is taken for granted in the printing industry. However it was not the case before the arrival of PostScript. Desktop publishing was a real nightmare with presence of innumerable document formats, font types, character encoding and image representation schemes. Today, PostScript is the industry standard for desktop publishing; it is flexible, extensible and can provide high-quality images – more than that, it is device independent! PostScript is not a file format – it is an object-oriented, interpreted language! Even in small hand-held printing devices, it is enough that the device has a PostScript interpreter - it can print the same documents used for professional publishing. That is the difference a language can make, even for printing documents. Yes, a language provides power and flexibility, but most importantly, it provides a means of abstraction – that is why it has a unique advantage over other schemes to get the work done.

Let us see another analogy from the field of education. English teachers find one major problem in teaching the language to non-native speakers in countries like India. Students tend to think in their native language and do word-by-word translation into English! Such mental 'transliteration' doesn't work for even gaining natural fluency if not gaining mastery over it. In just the opposite way, we write software that is essentially English, and put a layer over that to 'transliterate' it - it is just unnatural! When a programming language directly supports program-ming in a regional language, the barrier itself is removed; and at the same time it becomes a natural and effective tool for programmers to code in their native language. This should explain why native Japanese users find Ruby language to be very convenient, natural and useful though there is wide availability of programming languages based on English with a Japanese.

From the brave vision that programming should be available universally, our language derives its name from the musical term 'Swaram' - since music is an universal language! Swa-ram, in fact derives its main idea and inspiration from the Ruby language. Ruby is an interpreted, scripting and object-oriented language with extensive support for Japanese script. The primary motivation for the designer of Ruby, Yukihiro Matsumoto, was to support Japanese for scripting, since he found that no good scripting language could help him for that purpose.

Design Considerations

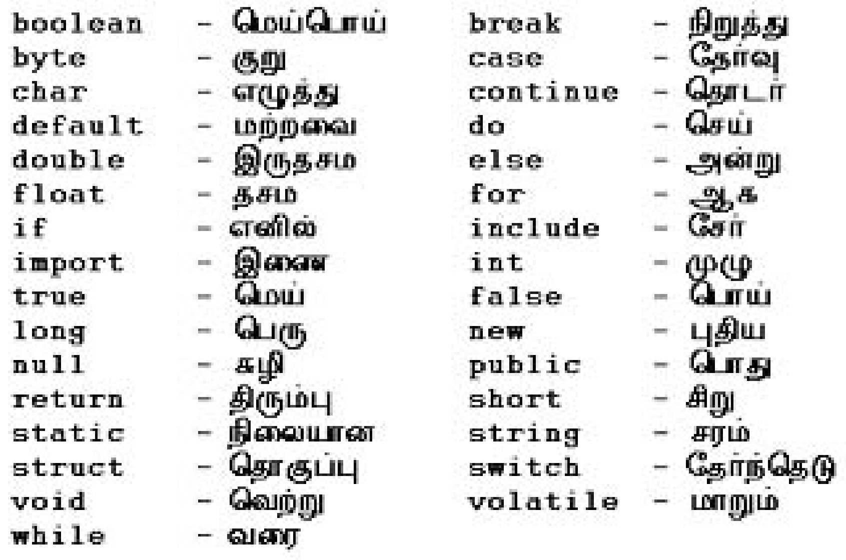
The objective of the authors was to design and implement a programming language and related tools especially for Tamil. The design considerations for such a language is as follows. Such a programming language:

1) Should be for general purpose and easy to use and learn
2) Should be platform independent and portable
3) It should be capable of having specialized support and functionality for various specific aspects of the Tamil language
4) Should use a well-established program translation technology and programming para-digm
5) Should be capable of supporting the Tamil language fonts directly and without much effort from the programmer
6) Should be easy to extend support to other Indian languages
7) Should be similar to widely used programming languages like C/C++/Java so that the learning curve will be minimal for those who already know programming and make the best use of it.

'Swaram' is thus designed as a general-purpose, easy-to-learn, platform-independent and port-able, both compiled and interpreted and extensible language.

Brief Technical Overview

Swaram derives most of its syntax and semantics from Java and C languages. A sample implementation has been developed from the scratch without using any compiler generator tools. The language has 30 keywords.



boolean	-	மெய்பொய்	break	-	நிறுத்து
byte	-	குறு	case	-	நேர்வு
char	-	எழுத்து	continue	-	தொடர்
default	-	மற்றவை	do	-	செய்
double	-	இருதசம	else	-	அன்று
float	-	தசம	for	-	ஆக
if	-	எனில்	include	-	சேர்
import	-	இணை	int	-	முழு
true	-	மெய்	false	-	பொய்
long	-	பெரு	new	-	புதிய
null	-	கழி	public	-	பொது
return	-	திரும்பு	short	-	சிறு
static	-	நிலையான	string	-	சரம்
struct	-	தொகுப்பு	switch	-	தேர்ந்தெடு
void	-	வெற்று	volatile	-	மாறும்
while	-	வரை			

List of Keywords in Swaram

Swaram supports PANDITHAM and Unicode coding schemes. To achieve extensibility, Swaram limits the language dependency to lexical analyzer only. The current implementation of the virtual machine is for the Windows platform. It has a very few APIs for the two languages that are supported - English and Tamil. A simple IDE (Integrated Development Environment) is also provided to facilitate programming in Tamil.

Program Translation

Swaram implementation comprises of three major parts:

1) A compiler for compiling the programs

2) A virtual machine (interpreter) to run it

3) Modules for compile-time and run-time support for Tamil

The compiler is written in C and the virtual machine in C++. The result of compiling a Swaram program is an intermediate file, which contains Java like bytecodes. Swaram bytecodes (intermediate language) are compatible with that of JVM's with a few exceptions (as discussed a bit later). A runtime interpreter loads the file and executes the bytecode instructions. In this way, Swaram follows Java technology for program translation.

Compilation

Swaram provides an option to choose between PANDITHAM or Unicode encoding scheme for the processing and storage of data. By default it uses PANDITHAM - this makes representation and processing of character data easier and basic requirements such as sorting becomes very easy. To achieve extensibility Swaram limits the language dependency to lexical analyzer only. Currently Swaram supports English and Tamil and is to be extended to support Hindi and Telugu. Whatever may be the source language, the same tokens are returned to the parser. The string information is passed as a PANDITHAM string. For our implementation we have Tamil source text in 'Ayyan', a PANDITHAM based font. A converter program has been implemented to store the text as PANDITHAM code. The converter program can be extended to support any font and language.

Intermediate File Format

Java's class file format is optimized only to Java and its object model. Swaram is capable of producing Java class files but for specialized use it has a separate file format - called the Swaram file format. It is very efficient, in that it is an improvement over Java class files. A better example is that the constant pool consists of 12 types of entries in Java Class File format. This creates an overhead in constant pool resolution and contributes in reducing runtime speed. But Swaram has only 4 types of constant pool entries that are direct and the remaining are included in the byte code itself resulting in faster execution.

Swaram Virtual Machine

The current implementation of SVM (Swaram Virtual Machine) runs under Win9x environment and can be implemented for any OS. The technology is as in JVM (Java Virtual Machine) but has more functionality for multilingual support and faster execution. SVM is a stack-oriented machine. The word size of the stack is 8 bytes. This means that all calculations are expandable to 8 bytes. Unlike JVM, SVM follows static linkage and supports procedural constructs. For faster execution SVM introduces new bytecodes like fipush, iipush, etc. Since strings are extensively used, they are special data types and have dedicated bytecodes like strload and strcat.

Compilation and Interpretation

Let us illustrate with a simple example of 'if' statement to show how the code is compiled and executed in Swaram.

```
If (A<B)
     ++A
```

It is compiled to the intermediate language as follows (comments added for explanation):

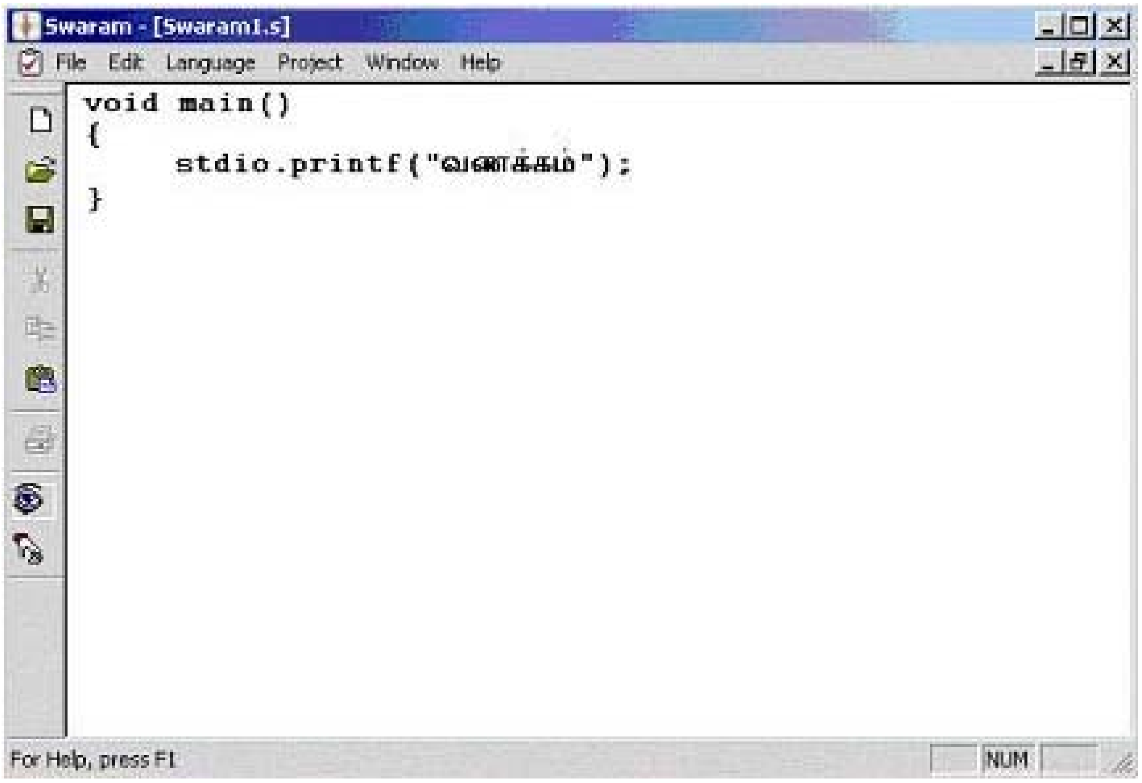
```
    iload_0      // push the first variable on the stack
    iload_1      // load the second variable on the stack
    if_icmpl T_0 // compare them and jump accordingly
F_0 :      // if the condition is false, it comes here
    iconst_0
    goto E_0
T_0 :      // if the condition is true
  iconst_1
E_0 :
    ifeq OUT_OF_IF_1
    iinc 0 1    // increment the first variable by one
OUT_OF_IF_1 :  // label to denote the end of if statement
```

The runtime interpreter provides a simulated environment for interpreting these bytecodes by allocating necessary memory and proving an operand stack for each method.

Programming In Swaram

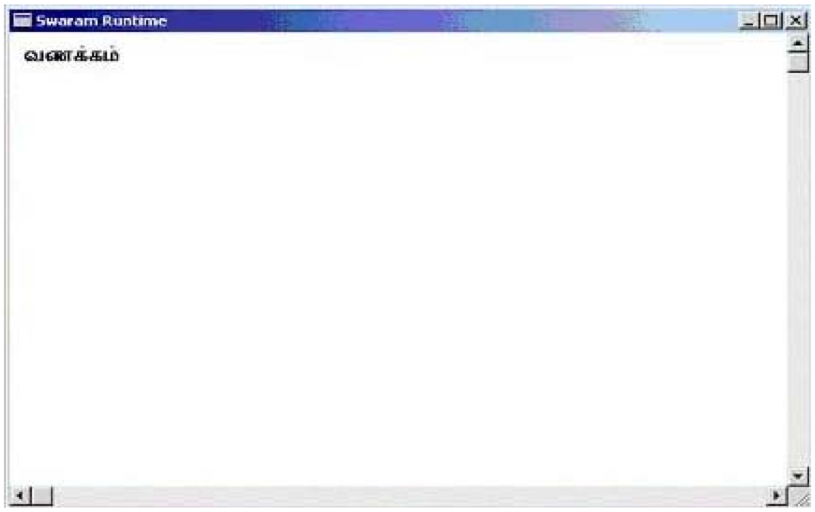
Swaram is designed such that it is easy for the beginners to learn and make the best use of the language. If a programmer already knows a language such as C/Java, he can straightaway start writing non-trivial programs in Swaram. Here are few simple examples to have a feel of programming in Swaram.

1) The canonical "Hello, world" program can be written in Swaram as follows:



```
void main()
{
      stdio.printf("வணக்கம்");
}
```

The output of the above program is:

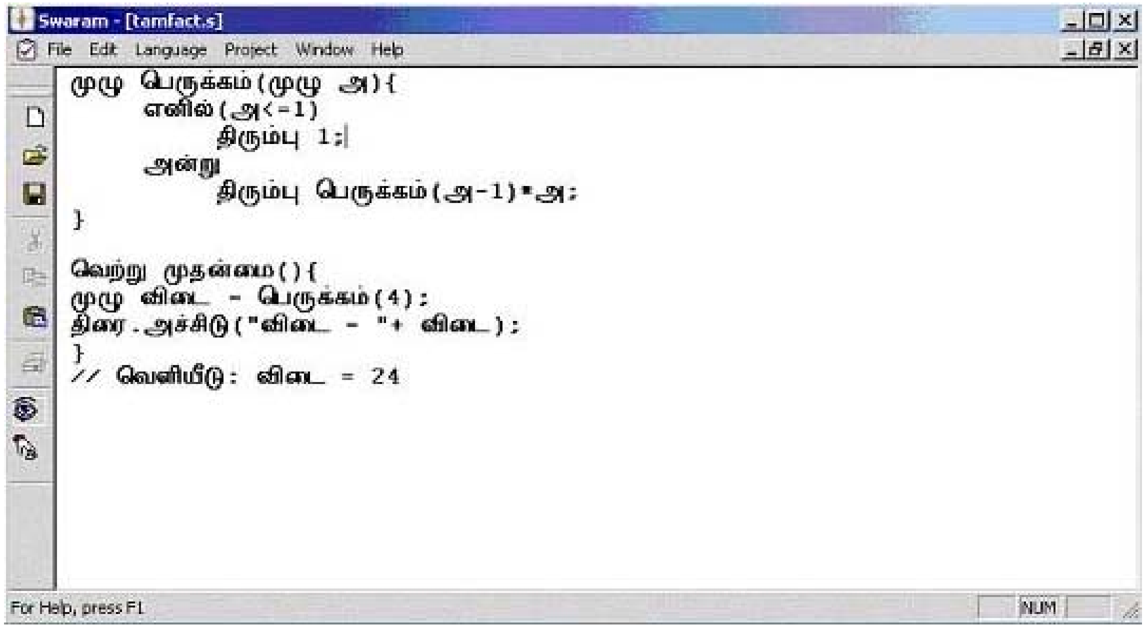


```
வணக்கம்
```

Similar to the popular languages like C, the program execution starts at the ‘main’ function. The APIs in Swaram are prefixed by the corresponding header file names. Currently three header files are supported: stdio (for IO), math (for mathematical operations) and graphics (for drawing simple figures).

Use the IDE to compile and execute the program that you have written by choosing the `Project->Compile` and `Project->Run` from the main menu of the Swaram editor. Any file name extensions can be given and .s is preferred.

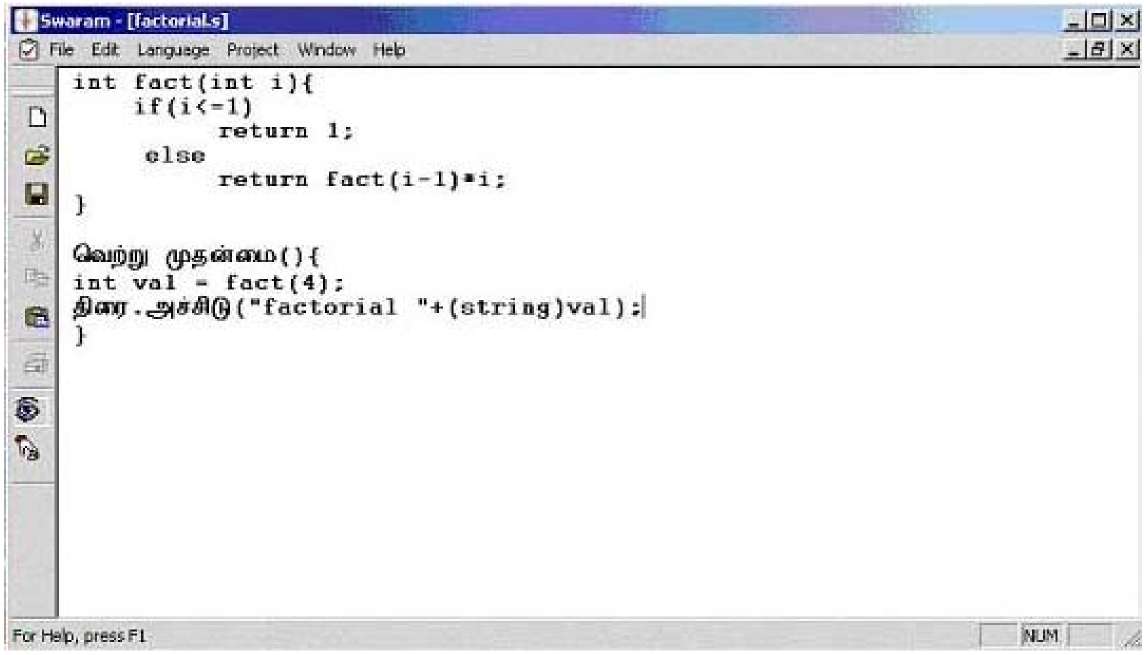
2) Program for finding the factorial of a number using recursion written in Tamil.



```
முழு பெருக்கம்(முழு அ){
        எனில்(அ<=1)
                திரும்பு 1;
        அன்று
                திரும்பு பெருக்கம்(அ-1)*அ;
}

வெற்று முதன்மை(){
முழு விடை = பெருக்கம்(4);
திரை.அச்சிடு("விடை = "+ விடை);
}
// வெளியீடு: விடை = 24
```

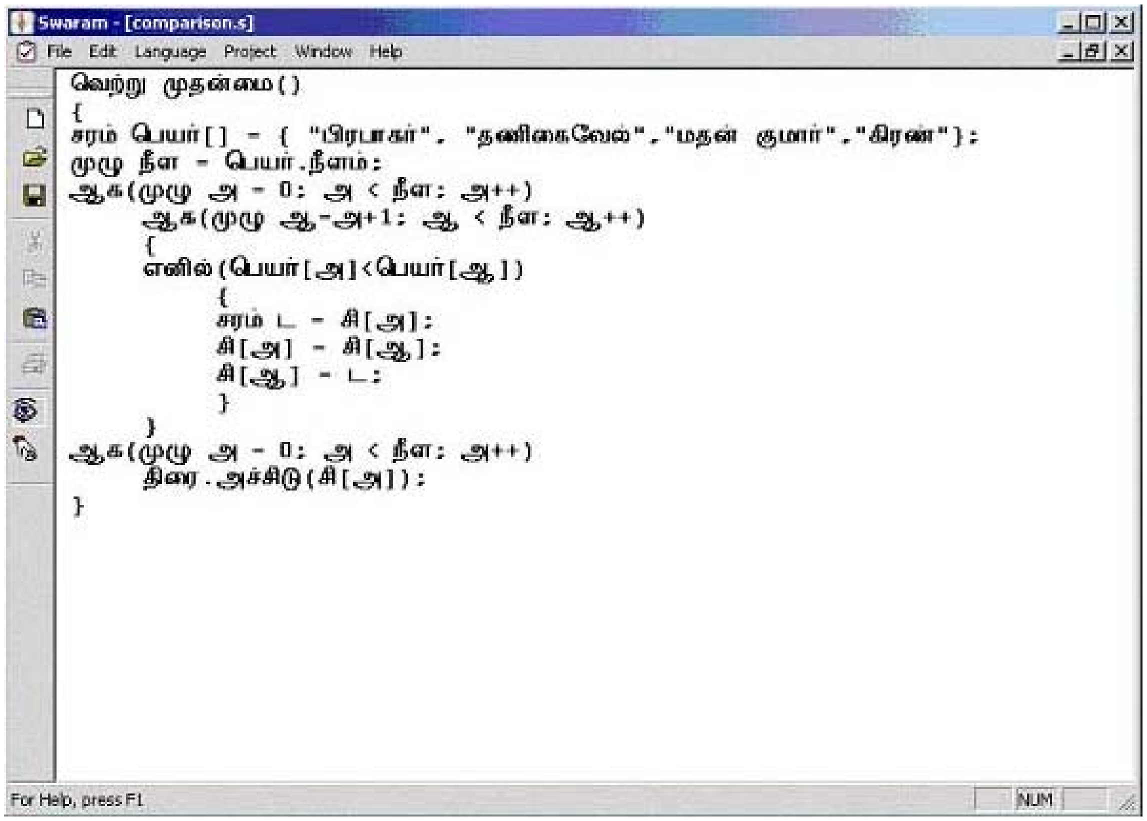
The below program depicts that you can mix English in between the Tamil ones.



```
int fact(int i){
     if(i<=1)
            return 1;
      else
            return fact(i-1)*i;
}

வெற்று முதன்மை(){
int val = fact(4);
திரை.அச்சிடு("factorial "+(string)val);
}
```

3) Program to sort the list of given names in Tamil



```
வெற்று முதன்மை()
{
சரம் பெயர்[] = { "பிரபாகர்", "தணிகைவேல்","மதன் குமார்","கிரண்"};
முழு நீள = பெயர்.நீளம்;
ஆக(முழு அ = 0; அ < நீள; அ++)
      ஆக(முழு ஆ=அ+1; ஆ < நீள; ஆ++)
      {
      எனில்(பெயர்[அ]<பெயர்[ஆ])
            {
            சரம் ட = சி[அ];
            சி[அ] = சி[ஆ];
            சி[ஆ] = ட;
            }
      }
ஆக(முழு அ = 0; அ < நீள; அ++)
      திரை.அச்சிடு(சி[அ]);
}
```

Conclusion:

'Swaram' is an attempt to give fillip for writing 'truly' Tamil software. It can also be a boon for students to learn programming in their mother tongue. A 'Proof of Concept' implementation has been successfully designed and a working version has been developed.

References:

Panditham	:	http://www.infitt.org/ti2000/tamilinaiyam/conference_hub.html
PostScript	:	http://www.adobe.com/products/postscript/pdfs/PLRM.pdf
Ruby	:	http://www.rubycentral.com or http://www.ruby-lang.org
Unicode	:	http://www.unicode.org

Ken Arnold and James Gosling, 'The Java Programming Language', The Java Series, Addison-Wesley, 1998.

Brian W. Kernighan and Dennis M. Ritchie, 'The C Programming Language', Second edition, Prentice-Hall, Inc., Englewood Cliffs, 1988.

Tim Lindholm and Frank Yellin, 'Java Virtual Machine Specification', The Java Series, Addison-Wesley, 1997.

Acknowledgements:

The authors would like to thank Dr P. Navaneethan, Asst Professor, Dept of MCA, PSG College of Technology, Coimbatore for his constant support and encouragement.

Authors:

S G Ganesh is a Software Engineer working for the C++ compiler front-end team in Hewlett-Packard, Bangalore, India. He is currently co-authoring 'Programming in C#, Java and C++: Comparing Best Features and Practices', for Addison-Wesley, USA. He is also the author of 'Deep C' (BPB Publications, ISBN 81-7656-501-6). His research interests include principles of programming languages and compiler design. He did his MCA degree from PSG College of Technology, Coimbatore, India. You can reach him at 's.g.ganesh@sify.com'.

G R Prakash is a Software Engineer working in Verizon, Chennai, India. He is currently co-authoring 'Programming in C#, Java and C++: Comparing Best Features and Practices', for Addison-Wesley, USA. His research interests include enterprise computing, operating systems, neural networks and microprocessor architectures. He did his MCA degree from PSG College of Technology, Coimbatore, India. You can reach him at 'grprakash@myrealbox.com'.

KK Ravikumar is a Software Engineer working in Verizon, Chennai, India. His research interests include Multilingual Computing, Object Oriented Programming and Data Structures. He did his MCA degree from PSG College of Technology, Coimbatore, India. You can reach him at 'kkravikumar@yahoo.com'.