CSCI 4160 Project 7

Due: see class calendar

Description:

This is optional individual project that build on the previous team project. In this project, you will implement an interpreter for Tiger language. To simplify the project, only int and string data types are supported.

Important Notes:

- The score of the project is to replace the lowest project you had for this course.

What to do in this project?

1. Make the following change to SymbolTable.h
   Replace the declaration of member function lookup by the following:

```
//Retrieve the value associated with the given lexeme in the hashtable list
//search starts at the given level blevel and ends at the global level
//if blevel is -1, start the search from the current level
//an exception is thrown if the lexeme doesn't exist
Entry& lookup(string, int blevel=-1);
```

2. Make the following change to SymbolTable.cpp
   Replace the implementation of member function lookup by the following:

```
template<class Entry>
Entry& SymbolTable<Entry>::lookup(string lexeme, int blevel)
{
	if ( blevel == -1)
		blevel = getLevel();

	//skip levels between blevel and getLevel()
	int		skip = getLevel() - blevel;
	for(Iterator it=tables.begin(); it!=tables.end(); it++)
	{
		//skip these levels
		if ( skip > 0 )
		{
			skip --;
			continue;
		}

		HashTable		&current = *it;
		HashTable::iterator	vit = current.find(lexeme);

		if ( vit != current.end() )
		{
			return current[lexeme]; //found the lexeme
		}
	}
	throw runtime_error("The given Lexeme " + lexeme + " doesn't
exist!") ;
}
```

3. If your TypeChecking.cpp file gives your trouble, please comment out the line 53 on main.cpp file.

4. What you need to do is to implement the following functions in Interpreter.cpp file. All other functions are provided by the instructor.

    - `const runtime::VarEntry& Interpreter::interp(const SimpleVar *v)`
    - `const runtime::VarEntry Interpreter::interp(const OpExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const VarExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const SeqExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const AssignExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const IfExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const WhileExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const ForExp *e)`
    - `const runtime::VarEntry Interpreter::interp(const LetExp *e)`

Classes in this project

The new classes introduced in this project are

- Class ActivationRecord: represents activation record for each function call.
- Class RuntimeStack: represents the runtime stack.
- Class Interpreter: represent interpreter for Tiger language.

Instructor provided files in the class repository

The following files are provided by the instructor:

- InterpreterProject folder. This folder contains a sample Visual Studio 2010 project. If you want to use this provided project for this assignment instead of creating your own project from scratch, please pay attention to the following:
- Description7.pdf: this file
- Rubric7.doc: the rubric used to grade this assignment.
- Test0.txt, test2.txt, and test2.txt. sample output for test0.tig, test1.tig and test2.tig

How to submit

1. Once you have finished, submit the project in the following way:
    - Copy the file projects/project4/rubric4.doc from the class repository to the project4 folder in your local repository of project4. Edit the file to put your name.
    - Commit the whole project7 folder to your local repository.
    - Push all the changes to master repository on ranger.
    - **Any commit of the project after the deadline is considered as cheating. If this happens, the latest version before the deadline will be graded, and you may receive up to 50 points deduction.**
2. You can also check your overall grade by update the rubric7.doc from the repository after the notice from the instructor.